DISEÑO Y DESARROLLO DE UN TRADUCTOR SINTÁCTICO ENTRE EL LENGUAJE DE DESCRIPCIÓN ARQUITECTURAL XADL 2.0 Y EL CÁLCULO PARA EL MODELAMIENTO FORMAL DE ARQUITECTURAS DE SOFTWARE $-\rho_{arq}$

DIANA CAROLINA DONOSO CASAS
20022020175
DAVID CAMILO SERRANO ABRIL
20022020130

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2008

DISEÑO Y DESARROLLO DE UN TRADUCTOR SINTÁCTICO ENTRE EL LENGUAJE DE DESCRIPCIÓN ARQUITECTURAL XADL 2.0 Y EL CÁLCULO PARA EL MODELAMIENTO FORMAL DE ARQUITECTURAS DE SOFTWARE $-\rho_{arq}$

DIANA CAROLINA DONOSO CASAS
20022020175
DAVID CAMILO SERRANO ABRIL
20022020130

Tesis de grado presentado para optar al título de Ingeniero de Sistemas

Director
HENRY ALBERTO DIOSA
Magister en Teleinformática

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2008

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

La ingeniería de sistemas es un área del saber en la que se pueden desarrollar investigaciones en diferentes campos. Uno de ellos es ingeniería de software, que a su vez incluye un sinnúmero de teorías, hipótesis e investigaciones que enmarcan una base sólida de conocimientos específicos en cuanto al desarrollo de software de alta calidad.

Dado que el mundo evoluciona a pasos agigantados, el desarrollo de software se hace cada vez más exigente y requiere de soluciones especializadas y a su vez muy complejas. El proceso de desarrollo ha dejado de ser una tarea netamente técnica y operativa y se ha convertido en todo un proceso de análisis, diseño y desarrollo de sistemas que soporten y den solución a problemas específicos, a través de metodologías que dan soporte al modelo de proceso.

Complementariamente, se ha abierto un espacio para la investigación de nuevos paradigmas de desarrollo de software que brindan beneficios adicionales como reducción del ciclo de desarrollo, mejora de la calidad del software, entre otros; siendo un ejemplo claro el desarrollo basado en componentes. A partir de este tipo de investigaciones, la complejidad de los sistemas software que están siendo desarrollados actualmente se ha incrementado notablemente, abriendo la puerta a nuevos problemas en la industria del software.

Más que problemas, estos pueden ser considerados oportunidades que permitan ampliar la base de conocimientos que enmarca el campo de la ingeniería de software. Específicamente, una de las áreas que se está empezando a trabajar fuertemente en este campo es la especificación formal de modelos arquitecturales de software, de lo cual se puede mencionar como resultados una serie de propuestas de Lenguajes de Descripción Arquitectural – LDA (del inglés

Architecture Description Languages ADLs), álgebras de procesos, métodos de especificación formal de arquitecturas de software, entre otros.

Sin embargo, aun existiendo investigaciones en este sentido, no existe una metodología unificada para la especificación de arquitecturas de software. Se ve entonces que por una parte, a través de los LDA's se puede hacer la descripción de arquitecturas de software, identificando sus componentes, conectores, configuraciones, etc., todo esto de manera estática; y por otro lado, a través de métodos de especificación formal se pretende describir su comportamiento, es decir, su descripción de manera dinámica.

Un trabajo interesante, y del que hace parte el producto de este proyecto, es aprovechar las virtudes que brindan tanto los LDA's como los métodos de especificación formal de arquitecturas de software, específicamente xADL 2.0 [Dashofy, 2007] y el cálculo – ρ_{arq}[Diosa et al., 2005], de los cuales se profundizará a lo largo de este documento.

A continuación se detallará a profundidad el proyecto propuesto, seguido de una fundamentación teórica de los tópicos que encierra el mismo, con el objetivo de enmarcar dentro de éste el producto presentado (Sección 4). Posteriormente, se presenta detalladamente todo el proceso de análisis y diseño desarrollados para el proyecto, identificando los modelos funcional, estructural y dinámico (Secciones 5, 6, 7, 8, y 9). En seguida se relacionan los aspectos involucrados en la implementación de los modelos definidos en los ítems anteriores así como los proyectos relacionados (Secciones 10 y 11). Finalmente, se presenta el proceso de construcción, validación y explicación de la aplicación desarrollada, estableciendo las conclusiones posteriores a la ejecución del proyecto y trabajos futuros (Secciones 12, 13 y 14).

2 GLOSARIO

ADL	Un ADL es un lenguaje descriptivo de modelado que se focaliza en la estructura de alto nivel de la aplicación antes que en los detalles de implementación de sus módulos concretos [Vestal, 1993]
XML	XML es el acrónimo de Lenguaje de marcado extensible. En realidad XML es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.
XSL	XSL es el acrónimo de lenguaje extensible en hojas de estilo, y es una familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio específico.

XSLT	XSLT, es una tecnología que implementa el lenguaje extensible en hojas de estilo. XSLT provee un entorno de trabajo para transformar un documento XML, donde se combina la entrada del documento XML con una XSL para obtener un documento nuevo de salida.
Parser (Interprete) XML	Un intérprete XML es un componente de software que puede leer y en la mayoría de los casos validar cualquier documento XML. Este intérprete vuelve los datos contenidos en un XML disponibles para una aplicación que necesite usar esos datos. [Gabrick y Weiss]
Arquitectura de Software	Una Arquitectura de Software es la representación de alto nivel de la estructura de un sistema o aplicación, que describe las partes que la integran, las interacciones entre ellas [Fuentes et al., 2001]. Una arquitectura de software tiene básicamente: componentes, conectores, enlaces y configuraciones.
Componente de Software	Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio [Szyperski, 1998].
Conector	Representan interacciones entre componentes. Corresponden a las líneas de las descripciones de caja y línea. Ejemplos típicos podrían ser tuberías

	(pipes), llamadas a procedimientos, difusión de eventos, protocolos cliente-servidor, o conexiones entre una aplicación y un servidor de base de datos. Estos también tienen una especie de interfaz que define los roles entre los componentes participantes en la interacción. [Reynoso y Kicillof, 2004]
Configuración	Existen ciertas configuraciones entre componentes y conectores ya predefinidas. Es similar a los patrones de diseño, que ya tienen una forma preestablecida al igual que algún objetivo en particular. Estos podrían llamarse patrones arquitecturales, aunque en algunas fuentes bibliográficas los relacionen también a estilos arquitecturales. Ya anteriormente se había mencionado algo sobre la existencia de arquitecturas basadas en capas, otras basadas en eventos, y así existen otros patrones arquitecturales que dada la naturaleza de este proyecto, no se explicaran al detalle [Buschmann et al., 1996].

3 JUSTIFICACIÓN

Una de las motivaciones principales que dieron origen a este trabajo, es la necesidad de culminar los estudios de pregrado en Ingeniería de Sistemas en la Universidad Distrital Francisco José de Caldas; pero no solo se busca obtener el título, sino también contribuir en el área de ingeniería de software, específicamente en investigación en el campo de arquitecturas de software.

Una gran variedad de problemas rodean la industria del desarrollo de software a nivel nacional, dentro de las cuales se destaca la falta de profundización en la investigación en el campo de arquitecturas de software. A nivel mundial se pueden encontrar varios proyectos de desarrollo de ésta índole, tales como ACME que es un lenguaje de intercambio de ADLs, desarrollado por Monroe & Garlan (Carnegie Mellon University) y Wile (University of Southern California); Darwin que es un ADL con énfasis en dinámica, desarrollado por Magee, Dulay, Eisenbach, Kramer; xADL que está basado en XML, desarrollado por Medvidovic, Taylor (University of California, Irvine; Universidad de California, Los Ángeles); cada uno de los cuales tiene un entorno de desarrollo asociado [Reynoso y Kicillof, 2004].

En Colombia los pocos avances alrededor de este tema son únicamente teóricos, más no existen desarrollos de software relacionados, lo que ha retrasado en gran medida la industria de software, que en la actualidad utiliza metodologías muy informales y no tiene en cuenta las nuevas teorías de desarrollo basado en arquitecturas, además que hay muy pocos aportes significativos al respecto.

Con este trabajo se contribuye con la investigación nacional sobre este tema, ya que toma una propuesta desarrollada a nivel nacional, como lo es el cálculo – ρ_{arq} y crea un puente con una tecnología externa de descripción de arquitecturas como

lo es xADL 2.0, haciendo así un aporte en la construcción de tecnología propia con respecto a la industria del software.

4 MARCO REFERENCIAL

En esta sección se describe el marco conceptual que soporta el desarrollo del proyecto. Está dividido en dos partes: la primera presenta la fundamentación teórica de los temas que abarca la investigación, como lo son el desarrollo basado en componentes, arquitecturas de software, métodos de especificación formal de arquitecturas de software y análisis sintáctico para la transformación ó traducción de gramáticas. La segunda parte presenta una explicación detallada de las tecnologías involucradas en el desarrollo del proyecto, tales como Lenguajes de descripción arquitecturales, xADL 2.0, XML y Esquemas XML y finalmente el cálculo – ρ_{arq}.

4.1 DESARROLLO BASADO EN COMPONENTES

En la actualidad, el rápido avance de las tecnologías, en lo que a software se refiere, ha requerido el surgimiento de nuevos paradigmas de programación que logren satisfacer esa velocidad y faciliten la labor de los programadores a la hora de implementar un diseño determinado. El ejemplo más claro de nuevos paradigmas de programación es la programación basada en componentes. La idea central de desarrollo de software basado en componentes (DSBC) es sentar las bases para el diseño y desarrollo de aplicaciones basadas en componentes de software reutilizables [Fuentes et al., 2001].

No obstante, para entender mejor la programación basada en componentes se debe hablar de ciertos elementos fundamentales de este nuevo paradigma de desarrollo de software. En primer lugar se define lo que es un sistema independientemente extendible [Szyperski, 1996]: Es un sistema que puede ser dinámicamente extendido y en donde pueden combinarse extensiones independientemente desarrolladas por distintas partes o entidades, sin conocimiento unas de otras. Así pues el desarrollo de software para un sistema de las características ya mencionadas y que además involucre comportamiento concurrente, genera una serie de problemas como la gestión de la evolución del software, el control de la vida de sus componentes ó la falta de visión global del sistema, entre otros. Con estos inconvenientes, se nota que la tradicional programación orientada a objetos podría tener dificultades a la hora de desarrollar sistemas de las características mencionadas. Como una respuesta inicial a estos nuevos sistemas y su alto nivel de complejidad surge la Programación Orientada a Componentes (POC), la cual es una extensión natural de la orientación a objetos que promueve el desarrollo y la utilización de componentes reutilizables dentro de lo que sería un mercado global de software [Szyperski y Pfister, 1997].

- Como ya se ha mencionado anteriormente, los problemas de la POO para satisfacer las necesidades de los nuevos sistemas complejos actuales, son limitantes para un desarrollador. La POC es una extensión natural de la POO que busca no solo resolver los problemas ya planteados sino que además aumente la facilidad de desarrollo y reutilización de componentes de software para posibilitar una mayor agilidad a la hora de diseñar e implementar un nuevo sistema de software. De acuerdo a lo anterior se muestra una definición de lo que es un componente:

  "*Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha*

de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio" [Szyperski, 1998].

Como complemento a lo anterior se puede decir que un componente tiene tres características propias:

- Un componente es una unidad independiente de despliegue: Esto quiere decir, que un componente desplegado en un sistema, solo es componente si en realidad es independiente del ambiente del sistema y de los demás componentes envueltos en el mismo, lo que significa que un componente debe ser capaz de encapsular sus rasgos constitutivos..
- Un componente es una unidad de composición con uno ó más componentes: Esto significa, que para que un componente logre interactuar con otros, debe en primer lugar tener sus servicios y necesidades bien definidos y especificados.
- Un componente no tiene estados observables externamente: Esta característica debe estar presente en todo componente, con el fin de evitar inconvenientes a la hora de la ejecución del sistema completo, donde el comportamiento del componente no debe ser modificado.

Una característica final recomendada para el trabajo con componentes en general, es evitar tener más de una copia de un componente en el sistema. Esto soportado en la idea de que al no tener un estado externo observable, el componente no podría ser distinguido de las copias del mismo, lo que causaría una pérdida de control sobre el sistema.

Con la breve descripción anterior es posible intuir la fortaleza del uso de componentes en el desarrollo de software. Si se tiene un componente que nos

presta un servicio bien definido que funciona en cualquier tipo de contexto y que además nos garantiza su disponibilidad frente a las conexiones que se podrían realizar con otros componentes, entonces lo que realmente se tiene es una unidad constitutiva del software que agilizaría cualquier desarrollo en particular. A pesar de esto, existe para los componentes una limitación significativa: solo pocos componentes son capaces de ejecutarse bajo ambientes con bajas garantías técnicas. Esto se refiere, a que solo en algunos casos un componente sería realmente independiente y que funcionaria en cualquier tipo de ambiente, pero en su gran mayoría, todos los componentes necesitan ejecutarse en ambientes previamente preparados para su funcionamiento. Un ejemplo podría ser un componente diseñado en lenguaje Java. Este componente solo podría ser ejecutado en un ambiente donde estuviera instalada previamente la máquina virtual.

Este es un campo que en la actualidad viene avanzando a pasos agigantados y cualquier problema que surja debe ser visto como una puerta más para la investigación y el mejoramiento del proceso de desarrollo de software. Como una consecuencia a futuro de esta nueva tendencia está la creación del mercado de componentes de software y el surgimiento de nuevas profesiones como los desarrolladores de componentes, arquitectos de componentes, certificadores de componentes, entre otros.

4.2 ARQUITECTURAS DE SOFTWARE

Dado el incremento de la complejidad de los sistemas de software actuales, los problemas de diseño van más allá de algoritmos y estructuras de datos de la computación. Siendo así, un nuevo problema en la industria del software es el diseño y la especificación del sistema completo, más cuando este involucra un nivel de complejidad alto. Una dificultad adicional, es que para realizar esa

especificación y ese diseño, no hay un claro estándar establecido, y diferentes organizaciones usan sus propias notaciones para realizar las especificaciones de su software. Entre estos ejemplos se pueden encontrar el International Standard Organization's Open Systems Interconnection Reference Model (una arquitectura para redes de trabajo en capas) [Paulk, 1985], el NIST/ECMA Reference Model (un entorno de ingeniería de software genérico que involucra una arquitectura basada en comunicación por capas) [Chen y Norman, 1992] [NIST, 1991], y el X Window System (una interfaz de usuario con manejo de ventanas que involucra una arquitectura basada en disparadores de eventos y callbacks)[Scheifler y Gettys, 1986]. De lo anterior se puede concluir que los logros alcanzados en el desarrollo basado en arquitecturas de software se basan en la identificación de patrones y estilos arquitecturales, más que en la definición de una taxonomía completamente aceptada de los paradigmas de arquitecturas.

Aún con lo anteriormente dicho no se ha aclarado cual es la más acertada definición de arquitectura de software. Se recurrirá al siguiente grafico para ejemplarizar el qué es y que debería tener una arquitectura de software.

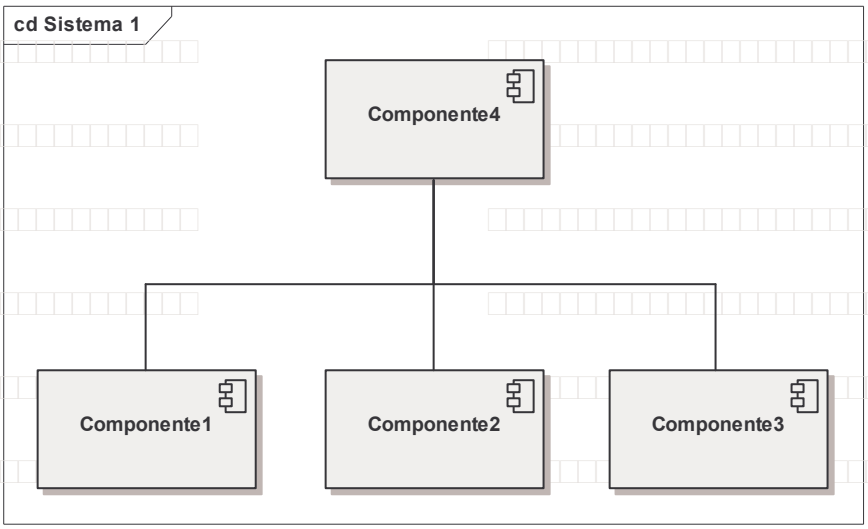

Figura 4.1: Típica y poco informativa presentación de una arquitectura de software

Se observa en la Figura 4.1, un ejemplo de arquitectura de software de un sistema no muy complejo. Se pueden decir algunas cosas de este diagrama, como que el sistema representado solo tiene cuatro componentes, que por la forma y alineación del diagrama el componente 4 tiene el control sobre los demás componentes y que además los otros componentes tienen cosas en común ya que se ubican al mismo nivel dentro del diagrama. También se puede suponer que los enlaces ó uniones entre componentes sirven para mantener el control del sistema, en donde puede que exista comunicación bidireccional, como puede que solo sea del nivel superior hasta el nivel donde están los componentes 1,2 y 3. Sin embargo este diagrama modificado en su forma podría representar para otro ingeniero de software otro tipo de sistema. Esto no solo reafirma las observaciones hechas anteriormente, sino que además muestra un claro ejemplo de las características que no debe tener una arquitectura de software.

Una arquitectura de software debe dar una cierta claridad sobre el sistema que representa, sin depender de la forma en si del grafo que describa dicha arquitectura ni tampoco de su gramática nativa. Algunas características del sistema, que deben ser mostradas a través de la arquitectura que lo representa son:

- La naturaleza de los componentes: En un determinado sistema, la naturaleza ó clase de los componentes determina en un gran porcentaje el entendimiento del sistema. Es muy diferente saber que en un sistema los componentes son simples JavaBeans que si son EJB[1]. Dependiendo

[1] Los JavaBeans son programas reusables que se desarrollan y pueden ser utilizados en cualquier aplicación que entienda este formato. Los componentes que se construyen sobre esta especificación son intraproceso que viven dentro de un espacio de direcciones único, y que, típicamente, se utilizan para manejar aspectos de la interfaz gráfica de usuario. La especificación de los EJB's, responde a una arquitectura más compleja. Los componentes construidos con esta especificación son interproceso, que viven en espacios de direcciones múltiples, como objetos distribuidos. Estos componentes son usados como objetos de negocio en aplicaciones transaccionales, y accedidos en forma remota.[TEK, 2008]

de cuál sea la naturaleza, se puede decir que el sistema usa comunicación remota con un servidor, o que en cambio es una aplicación centrada en una sola máquina. Por lo anterior, es fundamental que una arquitectura de software, sea capaz de mostrar la naturaleza de los componentes del sistema, ya sea en el grafo que la representa o en su gramática nativa.

- Significancia de los conectores: Como se mencionó anteriormente, la Figura 4.1 no dice nada acerca de cómo funcionan los conectores de la arquitectura representada. Una arquitectura debe ser capaz de mostrar el flujo de control y de datos de un sistema.

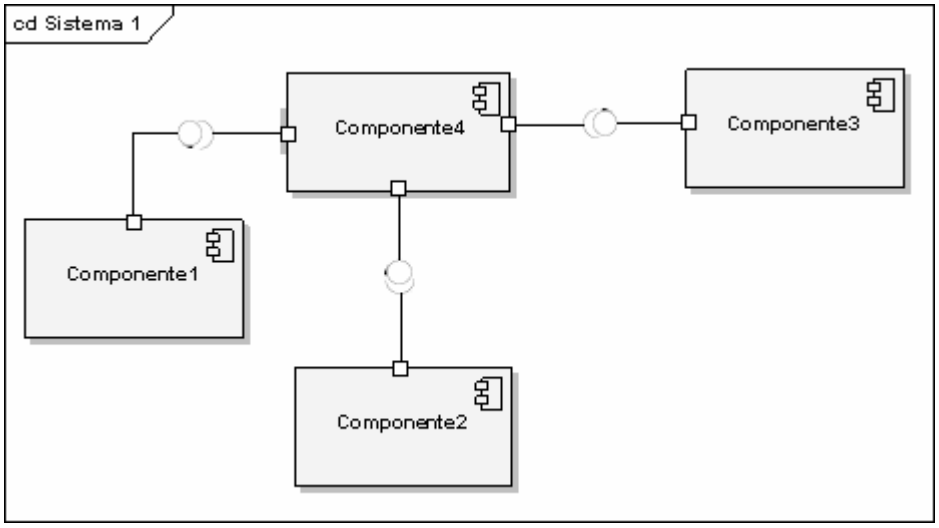

Figura 4.2: Arquitectura con el uso de la notación de componente

En la Figura 4.2, se aprecia una arquitectura similar a la presentada anteriormente, con la diferencia que ahora se uso la notación de componentes usada en UML. Esta notación nos dice que el componente 4 provee un cierto servicio a los componentes 1 y 3, pero requiere que el componente 2 le preste algún servicio para lograr su objetivo final. Sin embargo, a pesar que ahora se sabe la dirección del flujo de datos en el

sistema, aun falta saber si todo sucede de manera concurrente o si el componente 1 es atendido primero que el 3. Esto también debe expresarse en la arquitectura que representa el sistema.

- Alineación ó configuraciones: Existen ciertas configuraciones entre componentes y conectores que ya están predefinidas. Es similar a los patrones de diseño, los cuales ya tienen una forma preestablecida al igual que algún objetivo en particular. Estos podrían llamarse patrones arquitecturales, aunque en algunas fuentes bibliográficas los relacionen también a estilos arquitecturales. Ya anteriormente se había mencionado algo sobre la existencia de arquitecturas basadas en capas, otras basadas en eventos, y así existen otros patrones arquitecturales que dada la naturaleza de este proyecto, no se explicarán al detalle [Buschmann et al., 1996].

Ahora bien teniendo en cuenta lo anteriormente expuesto, se puede exponer una definición bastante acertada para lo que es una arquitectura de software:

> "*Una arquitectura de software es la estructura o estructuras de un sistema que comprende sus componentes, las propiedades visibles de esos componentes y las relaciones entre ellos".* [Bass et al., 1998].

Esta definición reafirma que los elementos que deben constituir la especificación estática de una arquitectura de software son: los componentes, los conectores y las configuraciones entre ellos.

4.3 LENGUAJES DE DESCRIPCIÓN ARQUITECTURAL (LDA)

Anteriormente ya se han dado algunas bases acerca de la nueva tendencia de desarrollo de software basada en componentes y a su vez en arquitecturas. Siguiendo esta línea evolutiva es necesario ahora exponer lo que son los LDA's: Lenguajes de descripción arquitecturales.

Dada la tendencia actual, los LDA's surgen como una herramienta de diseño y manejo de las arquitecturas de software que facilitan el entendimiento entre desarrolladores y por ende se aumenta el nivel de producción en la industria del software. Sin embargo, al igual que las arquitecturas del software y sus diferentes notaciones, los LDA's también tienen ciertas dificultades generadas por el hecho que diferentes organizaciones han producido su propio LDA, para su uso particular. Tan es así, que existen un número aproximado de 25 ó más LDA's a nivel mundial, los cuales difieren en notación grafica y gramática. A continuación se muestran los LDA's de más reconocimiento ante la comunidad internacional [Reynoso y Kicillof, 2004]:

ADL	Fecha	Investigador - Organismo	Observaciones
Acmé	1995	Monroe & Garlan (CMU), Wile (USC)	Lenguaje de intercambio de ADLs
Aesop	1994	Garlan (CMU)	ADL de propósito general, énfasis en estilos
ArTek	1994	Terry, Hayes-Roth, Erman (Teknowledge, DSSA)	Lenguaje específico de dominio - No es ADL
Armani	1998	Monroe (CMU)	ADL asociado a Acme
C2 SADL	1996	Taylor/Medvidovic (UCI)	ADL específico de estilo

ADL	Fecha	Investigador - Organismo	Observaciones
CHAM	1990	Berry / Boudol	Lenguaje de especificación basado en una analogía de soluciones químicas.
Darwin	1991	Magee, Dulay, Eisenbach, Kramer	ADL con énfasis en dinámica
Jacal	1997	Kicillof, Yankelevich (Universidad de Buenos Aires)	ADL - Notación de alto nivel para descripción y prototipado
LILEANNA	1993	Tracz (Loral Federal)	Lenguaje de conexión de módulos
MetaH	1993	Binns, Englehart (Honeywell)	ADL específico de dominio
Rapide	1990	Luckham (Stanford)	ADL & simulación
SADL	1995	Moriconi, Riemenschneider (SRI)	ADL con énfasis en mapeo de refinamiento
UML	1995	Rumbaugh, Jacobson, Booch (Rational)	Lenguaje genérico de modelado - No es ADL
UniCon	1995	Shaw (CMU)	ADL de propósito general, énfasis en conectores y estilos
Wright	1994	Garlan (CMU)	ADL de propósito general, énfasis en comunicación
xADL	2000	Medvidovic, Taylor (UCI, UCLA)	ADL basado en XML

Tabla 4.1: LDA's más conocidos a nivel mundial

Ahora bien, teniendo en cuenta lo que se expresó respecto a lo que es una Arquitectura de software, se puede dar una definición de lo que es un LDA:

"*Un LDA es un lenguaje descriptivo de modelado que se focaliza en la estructura de alto nivel de la aplicación antes que en los detalles de implementación de sus módulos concretos"* [Vestal, 1993]

Además de lo anterior, se puede agregar, que los LDA's, deben por lógica permitir la especificación clara de los elementos de una arquitectura de software: componentes, conectores y configuraciones entre ellos. Como complemento a lo ya dicho, y como consecuencia de la evolución de las herramientas CASE, es necesario decir que la mayoría de LDA's deben soportar el uso de este tipo de herramientas para diseñar, analizar, refinar e implementar una arquitectura de software, ya sea de manera grafica o con el uso de gramáticas simbólicas.

Teniendo en cuenta lo expuesto acerca de la nueva tendencia de desarrollo basado en arquitecturas de software, y el uso de LDA's para facilitar su especificación, se muestra a continuación una lista de definiciones claves a la hora de entender este nuevo paradigma [Reynoso y Kicillof, 2004]:

- Componentes: Representan los elementos computacionales primarios de un sistema. Intuitivamente, corresponden a las cajas de las descripciones de caja y línea de las arquitecturas de software. Ejemplos típicos serían clientes, servidores, filtros, objetos, pizarras y bases de datos. En la mayoría de los LDA los componentes pueden exponer varias interfaces, las cuales definen puntos de interacción entre un componente y su entorno.

- Conectores. Representan interacciones entre componentes. Corresponden a las líneas de las descripciones de caja y línea. Ejemplos típicos podrían ser tuberías (*pipes*), llamadas a procedimientos, difusión de eventos, protocolos cliente-servidor, o conexiones entre una

aplicación y un servidor de base de datos. Los conectores también tienen una especie de interfaz que define los roles entre los componentes participantes en la interacción.

- Interfaces: Son los puntos de salida y de entrada para los componentes y conectores. En general cada conector tiene interfaces hacia los componentes que prestan un servicio y hacia los componentes que reciben el servicio.

- Configuraciones o sistemas. Se constituyen como grafos de componentes y conectores. En los LDA's más avanzados la topología del sistema se define independientemente de los componentes y conectores que lo conforman. Los sistemas también pueden ser jerárquicos: componentes y conectores pueden subsumir la representación de lo que en realidad son complejos subsistemas.

- Propiedades. Representan información semántica sobre un sistema más allá de su estructura. Distintos LDA's ponen énfasis en diferentes clases de propiedades, pero todos tienen alguna forma de definir propiedades no funcionales, o pueden admitir herramientas complementarias para analizarlas y determinar, por ejemplo, el "*throughput*" y la latencia probables, o cuestiones de seguridad, escalabilidad, dependencia de bibliotecas o servicios específicos, configuraciones mínimas de hardware y tolerancia a fallas.

- Restricciones. Representan condiciones de diseño que deben acatarse incluso en el caso que el sistema evolucione en el tiempo. Restricciones típicas serían restricciones en los valores posibles de propiedades o en las configuraciones topológicas admisibles. Por ejemplo, el número de clientes que se puede conectar simultáneamente a un servicio.

- Estilos. Representan familias de sistemas, un vocabulario de tipos de elementos de diseño y de reglas para componerlos. Ejemplos clásicos serían las arquitecturas de flujo de datos basados en grafos de tuberías (*pipes*) y filtros, las arquitecturas basadas en pizarras con un espacio de datos compartido, o los sistemas en capas. Algunos estilos prescriben un marco de trabajo, un estándar de integración de componentes, patrones arquitectónicos o como se lo quiera llamar.

Como complemento final, se debe tener en cuenta que todo LDA debe soportar procesos de evolución del sistema que representa, para poder luego derivar subtipos a partir de los componentes existentes, y luego poder implementar esos subtipos en otros sistemas sin importar el contexto.

4.4 MÉTODOS DE ESPECIFICACIÓN FORMAL PARA ARQUITECTURAS DE SOFTWARE

Como ya se ha visto, para realizar la especificación de una arquitectura de software, es necesario usar algún tipo de lenguaje. Una opción son los LDA's, los cuales ya se han explicado de manera breve. Las especificaciones formales, son métodos que buscan mediante el uso de simbología matemática describir el comportamiento de un sistema. La mayoría de estos métodos fueron diseñados para modelar concurrencia en un sistema determinado y son más conocidos como algebras de procesos:

> *"Las álgebras de procesos surgen de la necesidad de dotar de un marco semántico formal a los lenguajes concurrentes, para los cuales las semánticas funcionales existentes hasta la fecha no eran adecuadas"* [Canal, 2000]

En general, todas las algebras de procesos tienen los siguientes elementos:

- Un conjunto de nombres de proceso.
- Un conjunto de nombres de eventos, canales o enlaces, que se utilizan para la sincronización y comunicación entre los procesos.
- Un conjunto de nombres de valores o datos, que son transmitidos a través de los canales.
- Una o varias constantes para representar los procesos inactivos o terminados.
- Acciones internas, que se definen como no observables, en el sentido de que no es posible determinar el momento en que el proceso lleva a cabo dichas acciones.
- Acciones de sincronización de procesos (en un evento) y de comunicación de datos entre dos procesos (por medio del envío y recepción de mensajes a través de un canal o enlace compartido por ambos).
- Operadores de restricción, que permiten delimitar el ámbito de un canal o enlace.
- Operadores de composición secuencial (entre procesos y de acciones con procesos), composición alternativa (no determinista) y composición paralela (tanto para el entrelazado de acciones como para la sincronización).
- Operadores para describir la composición alternativa determinista de procesos, por medio de una condición que determine la elección de una u otra alternativa.

Entre las algebras de procesos más conocidas están el Calculus of Concurrent Systems (CCS) [Milner, 1989], el ACP (Algebra of Communicating Processes),

desarrollada por Bergstra y colaboradores a inicios de los ochenta [Bergstra y Klop, 1985], el CSP (Communicating Sequential Processes) [Hoare, 1985] y el cálculo π [Milner, 1999] del cual se deriva el cálculo – ρ, y por ende el cálculo –ρ_{arq}, los cuales serán tratados más adelante.

4.5 ANÁLISIS DE GRAMÁTICAS

Hasta aquí, se ha descrito la parte teórica de lo que tiene que ver con el desarrollo de software basado en Arquitecturas y todo el marco conceptual que este tema requiere. Para este proyecto en particular, para lograr cumplir los objetivos propuestos, es necesario exponer un tema que se aleja un poco de lo que se venía exponiendo hasta ahora: El análisis de gramáticas.

Uno de los objetivos propuestos es construir un traductor entre la sintaxis de xADL2.0 y el cálculo –ρ_{arq}, para esto, es necesario conocer la teoría de análisis sintáctico existente. Este tema es mucho más conocido en ámbitos de compiladores, es de esto precisamente de lo cual se va hablar a continuación. La idea no es exponer todo lo relacionado a compiladores, sino simplemente dar un esbozo describiendo los componentes de un compilador, el entorno en el que trabajan los compiladores y algunas herramientas de software que facilitan la construcción de compiladores y tomar de ello lo que haga falta para desarrollar el proyecto planteado.

Básicamente un compilador es un traductor entre un lenguaje fuente y un lenguaje objeto. Lo que hace el programa es tomar como entrada algún tipo de fuente en un lenguaje determinado, luego mediante un proceso que ya se explicará más adelante, arroja como salida un objeto, que estará escrito en otro lenguaje. Además de esto, un compilador, puede arrojar errores de la

fuente ingresada los cuales evitan que la compilación sea realizada con éxito y que posiblemente el objeto de salida esté defectuoso.

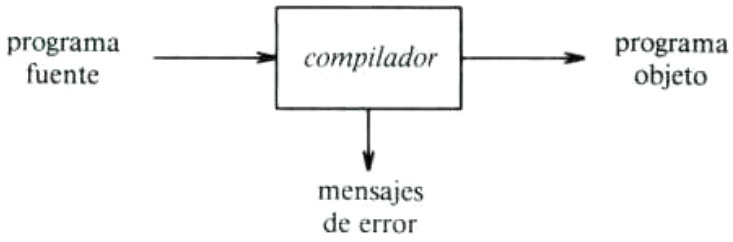

Figura 4.3: Compilador típico.

El proceso de compilación tiene dos grandes fases, el análisis y la síntesis. La parte del análisis divide al programa fuente en sus elementos componentes y crea una representación intermedia del programa fuente. La parte de la síntesis construye el programa objeto deseado a partir de la representación intermedia.

El análisis en el proceso de compilación consta de tres fases:

- Análisis lineal o léxico en el que la cadena de caracteres que constituye el programa fuente se lee de izquierda a derecha y se agrupa en componentes léxicos, que son secuencias de caracteres que tienen un significado colectivo

- Análisis jerárquico ó sintáctico, en el que los caracteres o los componentes léxicos se agrupan jerárquicamente en colecciones anidadas con un significado colectivo.

- Análisis semántico, en el que se realizan ciertas revisiones para asegurar que los componentes de un programa se ajustan de un modo significativo.

Lo anterior describe el proceso básico de compilación, sin embargo la Figura 4.4 nos dará algunos elementos ó fases extras de la estructura y comportamiento de un compilador.

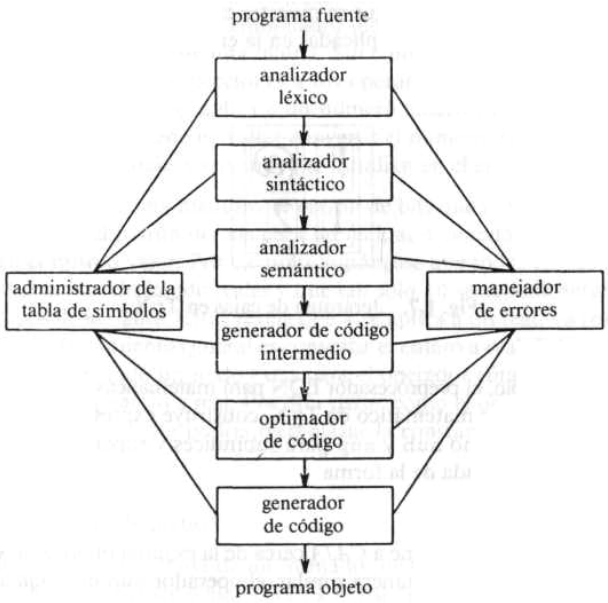

Figura 4.4 Fases ó elementos de un compilador

Se ven otras dos actividades, la administración de la tabla de símbolos y el manejo de errores, las cuales se muestran en interacción con las seis fases de análisis léxico, análisis sintáctico, análisis semántico, generación de código intermedio, optimización de código y generación de código. A continuación se explicará de manera breve las fases de compilación aun no estudiadas [Aho et al., 1998].

- Administrador tabla de símbolos: Una función esencial de un compilador es registrar los identificadores utilizados en el programa fuente y reunir información sobre los distintos atributos de cada identificador. Estos atributos brindan información acerca de la cantidad de memoria asignada a cada identificador, su tipo, su ámbito y en el caso de métodos estos atributos dicen el número de parámetros de entrada, el tipo de esos parámetros y los parámetros de retorno si es que hay un retorno. Para el manejo de estos datos se crea una estructura de datos llamada tabla de símbolos la cual contiene un registro por cada identificador, con los campos para los atributos del identificador.

- Manejador de errores: En particular, cada fase de compilación debe poseer una conexión al manejador de errores, ya que en cada fase es muy posible que se generen errores. Sin embargo las fases donde se detectan la mayoría de errores son el análisis léxico, el sintáctico y el semántico. En el primero y segundo se verifican la sintaxis general del programa fuente, los nombres de los identificadores, la finalización de las líneas de código, etc. Es necesario decir, que el compilador debe verificar todos los errores posibles en cada fase hasta llegar a la generación del código objeto. Esto significa que un compilador que se detenga en el primer error encontrado no sería útil. Ahora retornando a la parte de los errores hallados en la parte de análisis semántico, se puede decir que allí se controla los tipos y significancias de las formas sintácticas encontradas en la fase anterior. Así cada fase se comunica con el manejador de errores y en cualquier caso se generara una salida exitosa o una salida que especifique los errores presentados.

- Generador de código intermedio: La generación de código intermedio, es una etapa que no está presente en todos los compiladores. Lo que se

busca allí, es generar una representación intermedia del programa fuente, que debe ser fácil de leer y fácil de traducir al programa objeto.

- Optimizador de código: En esta etapa se busca que el código intermedio obtenido anteriormente, sea aún más fácil de leer y traducir al código objeto salida. Para lograr este objetivo, se simplifica el código intermedio lo más posible. Simplificar el código significa que las conversiones no necesarias deben eliminarse, las asignaciones a variables temporales innecesarias deben quitarse, etc. En el caso del proyecto planteado en este documento, tenemos que tener en cuenta que nuestro código fuente, está escrito en XML y pertenece a xADL2.0 y nuestro código objeto salida está escrito con simbología matemática correspondiente al cálculo – ρ_{arq} para el modelamiento formal de arquitecturas, por eso más adelante contemplaremos su sintaxis y estructura básica.

- Generador de código objeto: En general, un compilador arrojaría un código en lenguaje de máquina, en el caso particular del proyecto planteado en este documento, el código objeto será un código con simbología matemática que se apega a la sintaxis propuesta en el caculo – ρ_{arq}. La Figura 4.5 muestra un ejemplo completo del proceso de compilación básico para la expresión.

$$resultado = cantidad1 + cantidad2 * 60$$

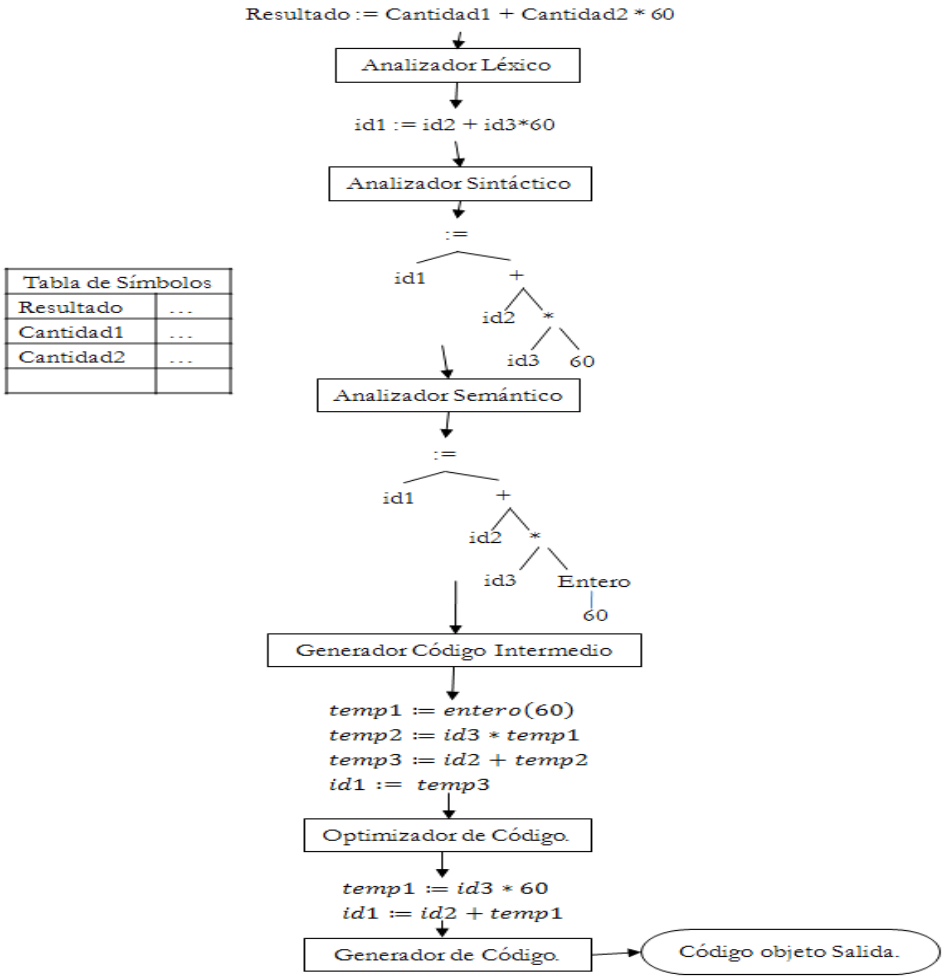

Figura 4.5: Proceso de compilación Básico.

4.6 CÁLCULO PARA MODELAMIENTO FORMAL DE ARQUITECTURAS DE SOFTWARE BASADAS EN COMPONENTES – ρ_{arq}

El cálculo $-\rho_{arq}$ se presenta como una solución al problema de la especificación de un modelo de referencia arquitectural de software a nivel de configuración, estructura y comportamiento [Diosa, 2005]. Este define un

vocabulario unificado que permite relacionar expresiones de configuraciones de especificaciones estáticas de arquitecturas de software. Para este proyecto, dichas especificaciones se describirán usando xADL 2.0.

El cálculo – ρ_{arq} es una aplicación del cálculo – ρ [ACSC, 1995], por lo que las entidades sintácticas usadas en el cálculo – ρ_{arq} son en su mayor parte las mismas del cálculo – ρ original, excepto las entidades de reacción interna, aplicación y la ausencia del concepto de celdas, al asumir una perspectiva puramente declarativa y observacional de los componentes.

El objetivo de éste cálculo de especificación formal, es que a partir de una configuración estática inicial que sólo es susceptible de pasar a una nueva configuración por medio de la aplicación de las reglas de la semántica operacional del cálculo – ρ_{arq}, se logre especificar los aspectos dinámicos de la arquitectura para posterior uso en asuntos de análisis de propiedades y detección de abrazos mortales [Diosa, 2005]. Al igual que otros cálculos para modelamiento formal, el cálculo – ρ_{arq} tiene definida una semántica operacional y unas entidades sintácticas presentadas en la siguiente tabla:

SÍMBOLOS	
x, y, z	variables
a, b, c	nombres
u, v, w ::= x|a	referencias

EXPRESIONES	
$E, F, G ::=$ T	Componente nulo
| $E \wedge F$	Composición
| $if(C_1 \ldots C_n)else$	Combinador de selección comprometida
| $x :: \bar{y}/E$	Abstracción
| $x\ \bar{y}/E$	Aplicación
| τ/E	Reacción interna
| $\exists wE$	Declaración
| $x : \bar{y}/E$	Replicación
$\phi, \psi ::= \dot{\top}$	Verdad lógica
| $\bot$	Falsedad lógica
| $x = y$	Restricción ecuacional
| $\phi \wedge \psi$	Conjunción de restricciones
| $\dot{\exists}\phi$	Cuantificador existencial

Fuente: [Diosa, 2005]

Tabla 4.2: Sintaxis del cálculo – ρ_{arq}

A continuación se presentan las características de la sintaxis del cálculo – ρ_{arq}, de acuerdo con lo expuesto en el documento de la referencia [Diosa et al., 2005]:

4.6.1 Símbolos

Se asume un alfabeto infinito de variables y un alfabeto infinito de nombres. Las variables son lugares para cargar nombres, es decir, no hay otros valores diferentes a los nombres. Tanto nombres como

variables son indiferenciadamente denominados referencias. Términos como $\tilde{x}$ equivalen a una secuencia como (*x1, x2, …, xn*).

4.6.2 Expresiones

En el cálculo – ρ_{arq} las expresiones representan Componentes y se simbolizan por *E, F, G*,….

- La expresión T se equipara a un componente nulo que no ejecuta acción alguna.
- La composición expresa ejecución concurrente de *E* y *F*.
- El combinador de selección comprometida o condicionada es una generalización útil del condicional, tiene la forma:
  if (C1) … (Cn) else G
  donde *Ck ::=* $\exists\tilde{x}$*(*$\emptyset$*k then Ek) con k=1* … n son argumentos.
- La abstracción por una sola vez representa recibir una entidad simbólica a lo largo de *x* que podrá reemplazar a $\tilde{y}$ en el componente E, siempre y cuando esta entidad simbólica recibida sea libre en el ámbito del componente E.
- La aplicación *x* $\tilde{y}$*/E* se interpreta como enviar $\tilde{y}$ a lo largo de *x* y continuar con la ejecución de *E*.
- La declaración ∃*wE* introduce una referencia *w* con alcance *E.*
- La replicación *x :* $\tilde{y}$*/E* se puede expresar de la forma: *x :* $\tilde{y}$*/E* ≡ *x ::* $\tilde{y}$*/E* ∧ *x :* $\tilde{y}$*/E* que permite instanciar componentes.
- Restricciones como $\emptyset, \psi$ pueden corresponder al valor de verdadero $(\dot{\top})$, al valor de falso $(\bot)$.
- Restricciones como $\emptyset, \psi$ pueden corresponder a restricciones ecuacionales $(x = y)$ con variables lógicas.

- Restricciones como $\emptyset, \psi$ pueden corresponder a conjunción de restricciones $(\emptyset \wedge \psi)$; la conjunción de restricciones es congruente a la composición de restricciones.
- La cuantificación existencial sobre restricciones es congruente a la declaración (restricción) de variables sobre restricciones $(\exists x \emptyset)$.

4.6.3 Semántica operacional

La semántica operacional define el comportamiento de una arquitectura y cambios en su estado a través de instrucciones que determinan operaciones entre los componentes de la misma. De esta manera, la semántica operacional para el cálculo – ρ_{arq} ha definido axiomas de congruencia estructural y reglas de reducción para controlar el comportamiento de componentes/conectores de la arquitectura en tiempo de ejecución. La tabla presenta los axiomas de congruencia estructural; variables ligadas son introducidas como argumentos formales de las abstracciones y por las declaraciones [Diosa et al., 2005].

(α – conversión)	Cambio de referencias ligadas por referencias libres
(ACI)	Λ es asociativa, conmutativa y satisface $E \wedge T \equiv E$
(Intercambio)	$\exists x \exists y E \equiv \exists y \exists x E$
(Alcance)	$\exists x\, E \wedge F \equiv \exists x\, (E \wedge F)\ si\ x \notin \mathcal{FV}(F)$
(Equiv.Restricciones)	$\emptyset \equiv \psi\ \ si\ \emptyset \Re_{\Delta} \psi\ y\ \mathcal{FV}(\phi) = \mathcal{FV}(\psi)$

Fuente: [Diosa et al., 2005]

Tabla 4.3: Congruencia Estructural Cálculo – ρ_{arq}

La tabla a continuación presenta las reglas de reducción que representan la semántica operacional. El operador de reemplazo $[\overline{x}/\overline{y}]$ requiere implícitamente que $\overline{x}$ y $\overline{y}$ tengan la misma longitud y que $\overline{y}$ sea lineal [Diosa et al., 2005].

$(\text{Å}_{\rho_{arq}})$ $\phi \wedge x{:}\overline{y}/E \wedge x'\overline{z}/F \longrightarrow \phi \wedge x{:}\overline{y}/E \wedge [\overline{z}/\overline{y}]\ E \wedge F \ si\ \phi \vDash_{\Delta} x = x', \mathcal{V}(\overline{z}) \cap \mathcal{BV}(E) = \emptyset$
$(\text{C}_{\rho_{arq}})\ \phi_1 \wedge \phi_2 \longrightarrow \psi \qquad si\ \phi_1 \text{ Å } \phi_2 \not\vdash_{\Delta} \psi$
$(\mathit{Then}_{\rho_{arq}})\ \phi \wedge \mathit{if}\ \psi\ \mathit{then}\ E\ \mathit{else}\ F\ \mathit{fi} \longrightarrow \phi \wedge E \qquad si\ \phi \vDash_{\Delta} \psi$
$(\mathit{Else}_{\rho_{arq}})\ \phi \wedge \mathit{if}\ \psi\ \mathit{then}\ E\ \mathit{else}\ F\ \mathit{fi} \longrightarrow \phi \wedge F \qquad si\ \phi \vDash_{\Delta} \neg\psi$

Fuente: [Diosa et al., 2005]

Tabla 4.4: Reglas de Reducción del Cálculo – ρ_{arq}

Estas tres reglas de reducción, desde la perspectiva arquitectural, se pueden asociar a restricciones del contexto global de la arquitectura de referencia y controlan (si son suficientemente fuertes) el comportamiento de componentes/conectores de la misma en tiempo de ejecución.

La regla de reducción de combinación de restricciones (Cp) merece especial atención porque restricciones a nivel global de la arquitectura podrían obtener más información o ampliarse por este medio.

4.6.4 Notación de componente

Para describir la notación usada por el cálculo Cálculo – ρ_{arq}, se tendrá en cuenta la Figura 4.6 que muestra como se representa un componente en UML y a partir de esta se explicará la especificación formal en el cálculo – ρ_{arq} para diferentes configuraciones.

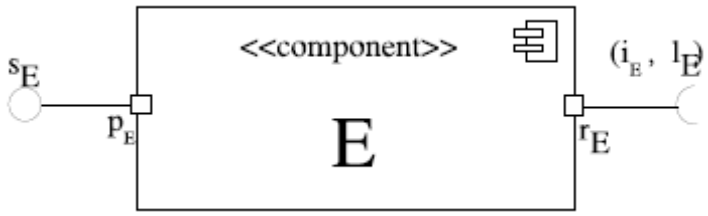

Figura 4.6: Notación gráfica de componente

Para el componente *E* que se presenta en la Figura 4.6, la interfaz de salida se denominará $PROV_e(p, s)$ y se definirá formalmente por [Diosa, 2005]:

$$PROV_E(p,s) \stackrel{def}{=} p_E : x/xs_E \equiv p_E :: x/xs_E \wedge p_E : x/xs_E$$

correspondiente a la entidad sintáctica **replicación** y se interpretaría como: *se espera a lo largo de* p_E *un valor de una locación a la cual se enviará el servicio* s_E. La interfaz de entrada se denominará $REQ_E(r,l)$ y se definirá por [Diosa, 2005]:

$$REQ_E(r,l) \stackrel{def}{=} r_E :: y/yl_E$$

que se puede interpretar como: se espera a lo largo de $r_E$un nombre de acceso a un servicio, que al ser aplicado a la ubicación l_E, conecte ésta con el servicio.

Teniendo definidas las interfaces de entrada y salida del componente de la Figura 4.6, se tiene entonces su representación desde un punto de vista observacional. Para este caso, la definición del componente E sería como:

$$E \stackrel{def}{=} PROV_E(p,s) \wedge REQ_E(r,l) \wedge l_E :: i_E/E$$

A continuación se presenta una figura (Figura 4.7) que muestra como es la notación gráfica del ensamble entre componentes, a partir de la cual se explicará la especificación formal de un conector entre dos componentes y de la misma configuración.

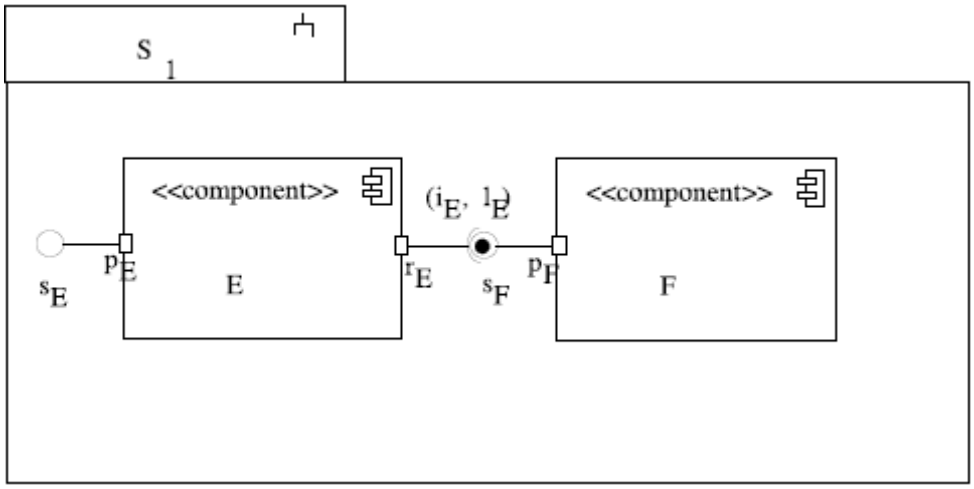

Figura 4.7: Notación gráfica de ensamble de componentes

Como se puede observar, se tienen dos componentes [Diosa, 2005]:

$$E \stackrel{def}{=} (p_E : x/xs_E) \wedge (r_E : y/yl_E) \wedge (l_E :: i_E/E)$$
$$E \stackrel{def}{=} (p_F : z/zs_F)$$

En este caso, para conectar activamente *E* y *F* se requiere un conector que logre dar acceso al servicio; este conector de ensamble se denominará C_{FE} y se especificará como [Diosa, 2005]:

$$C_{FE} \stackrel{def}{=} r_E\, p_F$$

Al actuar en forma concurrente (composición) hará que el sistema S1 representado por esta configuración inicie una dinámica de ejecución que se puede especificar así:

$$S_1 = E \wedge F \wedge C_{FE}$$
$$= \{(p_E : x/xs_E) \wedge (r_E : y/yl_E) \wedge (l_E :: i_E/E)\} \wedge \{(p_F : z/zs_F)\} \wedge \{r_E p_F\}$$

Ahora si se aplican los axiomas de congruencia estructural y las reglas de reducción del cálculo – $\boldsymbol{\rho_{arq}}$ sin presencia de restricciones, se tiene:

$$S_1 \xrightarrow{A_{\rho arq}} \{(p_E : x/xs_E) \wedge ([p_F/y]yl_E) \wedge (l_E :: i_E/E)\} \wedge \{(p_F : z/zs_F)\}$$
$$\equiv \{(p_E : x/xs_E) \wedge p_F l_E \wedge (l_E :: i_E/E)\} \wedge \{(p_F : z/zs_F)\}$$
$$\xrightarrow{A_{\rho arq}} \{(p_E : x/xs_E) \wedge (l_E :: i_E/E)\} \wedge [l_E/z]zs_F$$
$$\equiv \{(p_E : x/xs_E) \wedge (l_E :: i_E/E)\} \wedge l_E s_F$$
$$\xrightarrow{A_{\rho arq}} p_E : x/xs_E \wedge [s_F/i_E]E$$

Con el cálculo – $\boldsymbol{\rho_{arq}}$ se puede también especificar el flujo de servicios de una arquitectura compleja, tal información puede ser profundizada en el documento de la referencia [Diosa, 2005].

4.7 XSLT

XSLT (del inglés eXtensible Stylesheet Language/Transform) es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros, incluso a formatos que no son XML. XSLT permite al autor de una hoja de estilo transformar un documento original XML de dos formas:

manipulando y ordenando el contenido (incluyendo una reordenación general de éste si es necesario), y transformando el contenido en distintos formatos.

Este lenguaje exige definir qué transformaciones se quieren hacer y a qué partes del documento. Para lo primero el propio XSLT define unos elementos, pero para lo segundo XSLT se basa en la sintaxis que define XPath. Por lo tanto, XSLT es un lenguaje que va a definir qué transformaciones se van a realizar y cada vez que se seleccione un elemento a modificar utilizará XPath. XPath son cadenas de expresiones regulares que hacen referencia a alguna estructura dentro del documento XML.

Para conseguir realizar la modificación es necesario escribir una hoja de estilo explicando, con la sintaxis apropiada, qué se desea que el procesador XSLT haga con el archivo XML. La hoja de estilo tiene la extensión 'xsl' y una vez definida funcionará con cualquier procesador XSLT escrito en cualquier lenguaje y ejecutado en cualquier entorno.

Lo que consiguen las hojas de estilo es separar la información (almacenada en un documento XML) de su presentación, usando en cada caso las transformaciones que sean necesarias para que el contenido aparezca de la forma más adecuada en el cliente. Adicionalmente, se pueden usar diferentes hojas de estilo, o incluso la misma, para presentar la información de diferentes maneras dependiendo de los deseos o de las condiciones del usuario. Otra de las ventajas es que permite separar el contenido de la presentación, permitiendo modificar aspectos visuales fácilmente sin que los contenidos se vean mezclados en el proceso.

Así mismo, XSLT contiene básicamente tres tipos de elementos [Cernuda, 2004]:

- Elementos de XSLT. Pertenecen al "*namespace"* xsl, y por tanto sus etiquetas llevan el prefijo **xsl**.
- Elementos LRE (del inglés Literal Result Elements). Consiste de cualquier elemento que no sea instrucción y que debe ser copiado tal cual al documento resultante.
- Elementos de extensión. Son elementos no estándar (al igual que los LRE), que son manejados por implementaciones concretas del procesador.

Las funciones XLST se utilizan como parte de las expresiones XPath de una hoja de estilos XSLT para tener acceso al nodo actual (current()), fusionar distintos archivos de datos XML en uno solo (document()), mantener compatibilidad de versiones (element-available() o function-available()), dar formato a los números (format-number()) o comprobar propiedades del sistema.

Existen en el mercado varios procesadores XSLT, algunos con licencia comercial, otros con licencia libre, los cuales permiten la edición de documentos XML libres de errores, ayudando así a validar estos documentos contra DTD's o Esquemas y forzando a ceñirse a una estructura XML válida. Entre los más conocidos se encuentran Xalan, SAXON, XMLSpy, [Urretavizcaya et al., 2007] entre otros.

En general, los aspectos más importantes de XSLT han sido explicados en éste apartado. Si desea obtener información más detallada acerca de cómo utilizar las herramientas que proporciona XSLT puede consultar la página del W3C [W3C, 2007]; de la misma manera, podrá consultar información acerca de todos los elementos y funciones XSLT en la página del Microsoft Developer Network [MSDN, 2007]

4.8 xADL 2.0

Hasta ahora se ha dado la fundamentación necesaria para entender lo que es un LDA, sin embargo también es necesario saber algo de XML, ya que la unión de estas dos tecnologías, más la utilización de esquemas XML [W3C, 1, 2007], ha dado como resultado, un lenguaje de descripción arquitectural basado en XML que además aprovecha los esquemas ya mencionados, para definir su estructura. No obstante, dado que XML es un lenguaje ampliamente difundido, no se profundizara en la descripción detallada del mismo. Sin embargo para una información completa y estructurada puede consultar a [Benz y Durant, 2003], [Floyd, 2000] y verificar la documentación del W3C en [W3C, 2, 2007]. A continuación se explicara de manera breve que es xADL2.0 y por qué se escogió este LDA y no otro.

Como su nombre lo indica, xADL2.0 es un LDA (Ver sección 4.3) que está basado en XML y usa esquemas para definir todos sus elementos. Así pues siendo un LDA, puede modelar cualquiera de las cuatro estructuras arquitecturales más comunes: Componentes, conectores, interfaces y las diferentes configuraciones entre ellos. Como está basado en XML es extensible, una característica que lo hace bastante útil a la hora de definir nuevas estructuras. Esta última razón justifica la razón de elegir este LDA y no otro.

xADL2.0 está basado en varios esquemas para definir sus estructuras, a continuación se verá un ejemplo de definición de un componente de manera gráfica y luego se estudiará el código XML generado, para este fin se usará XArchStudio4 [UCI, 2007]:

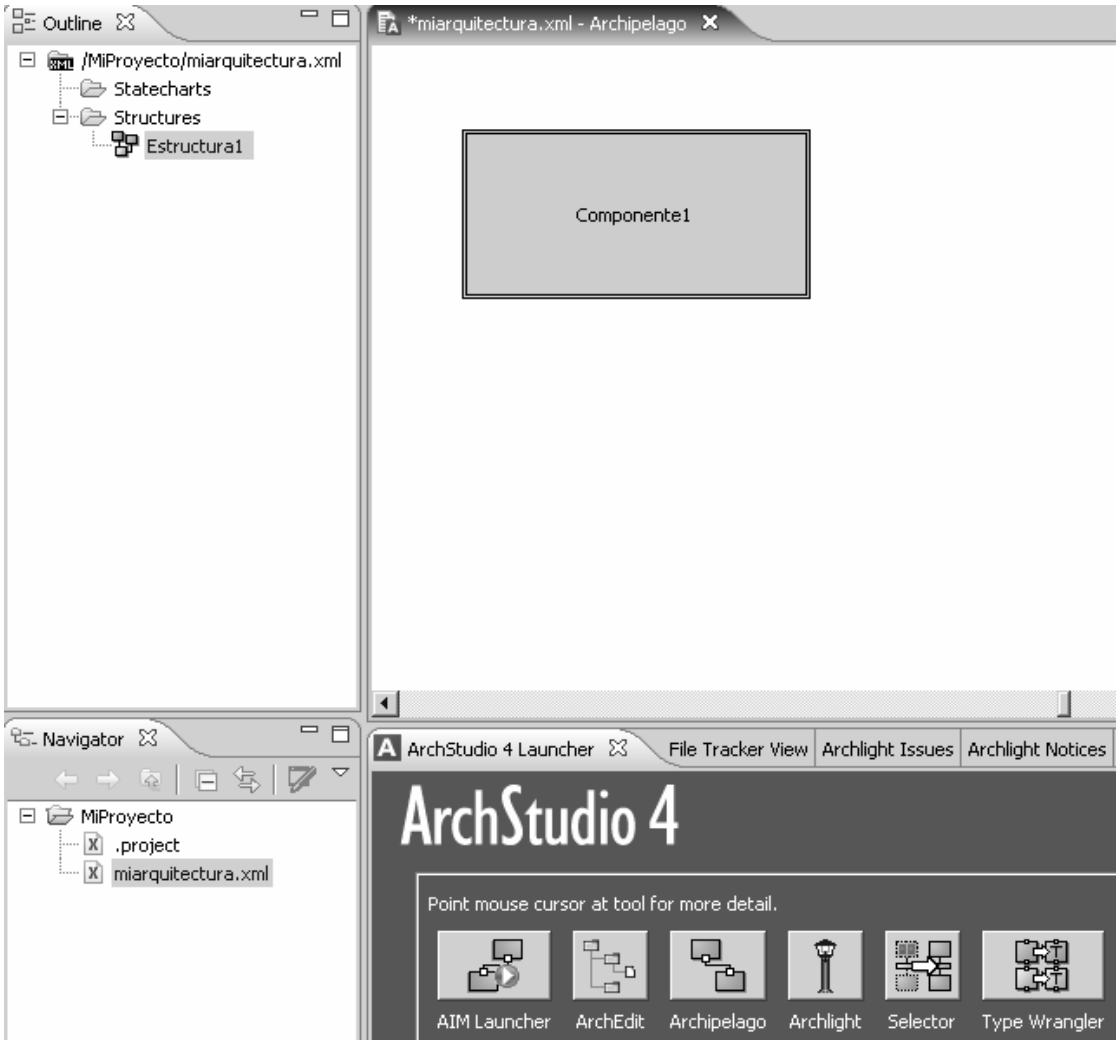

Figura 4.8: Ejemplo gráfico de componente XADL2.0 en ArchStudio4

La Figura 4.8 muestra la definición de un componente simple, dentro de una nueva estructura llamada Estructura1, perteneciente a una arquitectura de nombre miarquitectura. Ahora veamos el código generado dentro del archivo XML:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <instance:xArch xmlns:instance="http://www.ics.uci.edu/pub/arch/xArch/instance.xsd" xmlns:hints3="
   http://www.ics.uci.edu/pub/arch/xArch/hints3.xsd" xmlns:statecharts="
   http://www.ics.uci.edu/pub/arch/xArch/statecharts.xsd" xmlns:types="http://www.ics.uci.edu/pub/arch/xArch/types.xsd"
    xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation
   ="http://www.ics.uci.edu/pub/arch/xArch/hints3.xsd http://www.isr.uci.edu/projects/xarchuci/ext/hints3.xsd
   http://www.ics.uci.edu/pub/arch/xArch/statecharts.xsd http://www.isr.uci.edu/projects/xarchuci/ext/statecharts.xsd
   http://www.ics.uci.edu/pub/arch/xArch/types.xsd http://www.isr.uci.edu/projects/xarchuci/ext/types.xsd
   http://www.ics.uci.edu/pub/arch/xArch/changesets.xsd http://www.isr.uci.edu/projects/xarchuci/ext/changesets.xsd">
3
4      <types:archStructure types:id="archStructureffa80004-0ebe135b-d8baf0d4-de850003" xsi:type="types:ArchStructure">
5          <types:description xsi:type="instance:Description">Estructura1</types:description>
6              <types:component types:id="componentffa80004-0ebeb596-9f3ee841-de850006" xsi:type="types:Component">
7                  <types:description xsi:type="instance:Description">Componente1</types:description>
8              </types:component>
9      </types:archStructure>
10 </instance:xArch>
```

Figura 4.9: Archivo XML generado con el componente grafico.

Se observan dos partes en el archivo. La primera parte es la línea dos. Allí se ven las diferentes referencias a las páginas web donde están los esquemas predefinidos y utilizados para crear arquitecturas de software con xADL2.0. La segunda parte corresponde a la definición del componente. Es clara la forma de anidamiento del componente dentro de la estructura y esta a su vez dentro de la arquitectura. Se ve que cada elemento dentro del documento, tiene un tipo, un id y una descripción. Sin embargo hasta ahora se ha creado un componente, veamos ahora el código generado con dos componentes unidos con un enlace:

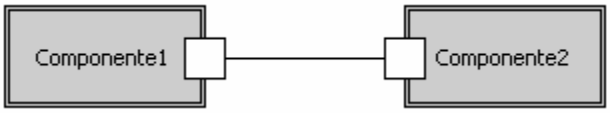

Figura 4.10: Ejemplo grafico de la unión de dos componentes.

```
3     <types:archStructure types:id="archStructureffa80004-0ebe135b-d8baf0d4-de850003" xsi:type=
"types:ArchStructure">
4         <types:description xsi:type="instance:Description">Estructura1</types:description>
5             <types:component types:id="componentffa80004-0ebeb596-9f3ee841-de850006" xsi:type="types:Component">
6                 <types:description xsi:type="instance:Description">Componente1</types:description>
7                     <types:interface types:id="interfaceffa80004-198d0e77-5c0047ab-3d56015a" xsi:type=
"types:Interface">
8                         <types:description xsi:type="instance:Description">interfaz1</types:description>
9                             <types:direction xsi:type="instance:Direction">none</types:direction>
10                    </types:interface>
11            </types:component>
12            <types:component types:id="componentffa80004-198c5f89-3eae5b23-3d5600d1" xsi:type="types:Component">
13                <types:description xsi:type="instance:Description">Componente2</types:description>
14                    <types:interface types:id="interfaceffa80004-198d5b20-7ab87865-3d560197" xsi:type=
"types:Interface">
15                        <types:description xsi:type="instance:Description">interfaz2</types:description>
16                            <types:direction xsi:type="instance:Direction">none</types:direction>
17                    </types:interface>
18            </types:component>
19            <types:link types:id="linkffa80004-198d7937-6356f5b1-3d56019f" xsi:type="types:Link">
20                <types:description xsi:type="instance:Description">enlace1</types:description>
21                    <types:point xsi:type="instance:Point">
22                        <instance:anchorOnInterface xlink:href="#interfaceffa80004-198d0e77-5c0047ab-3d56015a"
xsi:type="instance:XMLLink" xlink:type="simple"/>
23                    </types:point>
24                    <types:point xsi:type="instance:Point">
25                        <instance:anchorOnInterface xlink:href="#interfaceffa80004-198d5b20-7ab87865-3d560197"
xsi:type="instance:XMLLink" xlink:type="simple"/>
26                    </types:point>
27            </types:link>
28    </types:archStructure>
29</instance:xArch>
```

Figura 4.11: Código generado por la unión de los dos componentes.

En este código se obvió el encabezado de referencia hacia los esquemas y la declaración inicial de XML. La definición de los componentes es similar a la mostrada en la Figura 4.9, sin embargo al agregar interfaces vemos que éstas tienen un atributo adicional: Dirección. La dirección en la comunicación de los componentes es muy diciente de cuál es el componente que presta servicio y a quien se los presta. También se ve que el elemento que describe un enlace es llamado “link”. Este “link” tiene dos elementos anidados los cuales son los puntos extremos, y son de tipo “Point”. Cada punto extremo, en este caso está asociado a una interfaz, es claro que para referenciar a que interfaz se conecta cada punto, el software no hace referencia a la descripción sino al identificador asignado a cada interfaz. Hay un atributo del “link” que es el tipo

de asociación a la interfaz el cual es en este caso simple. A continuación un último ejemplo que incluye un elemento conector:

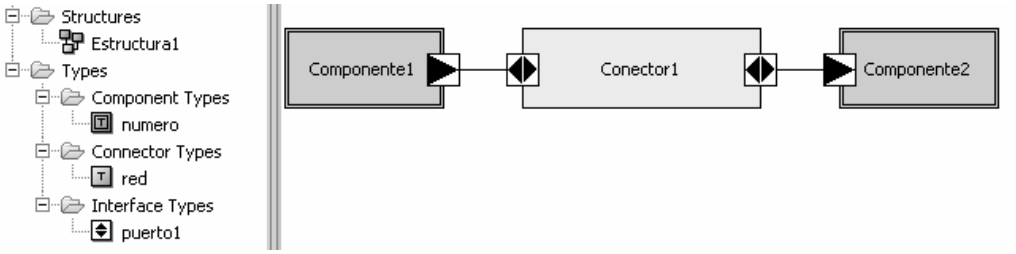

Figura 4.12: Configuración sencilla entre componentes y un conector.

En la Figura 4.12, no solo se observa la configuración entre un conector y dos componentes, también se observa al lado izquierdo que en xADL2.0 es posible también definir los tipos de componentes, conectores ó interfaces. El archivo XML generado es el siguiente:

```
18        <types:connector types:id="connectorffa80004-19e1430b-2753e671-3d560690" xsi:type="types:Connector">
19            <types:description xsi:type="instance:Description">Conector1</types:description>
20                <types:interface types:id="interfaceffa80004-19e15124-fa0bce59-3d5606b2" xsi:type=
    "types:Interface">
21                    <types:description xsi:type="instance:Description">interfaz2</types:description>
22                        <types:direction xsi:type="instance:Direction">inout</types:direction>
23                </types:interface>
24                <types:interface types:id="interfaceffa80004-19e15b36-759c5bfc-3d5606b9" xsi:type=
    "types:Interface">
25                    <types:description xsi:type="instance:Description">interfaz3</types:description>
26                        <types:direction xsi:type="instance:Direction">inout</types:direction>
27                </types:interface>
28        </types:connector>
```

Figura 4.13: Definición de un conector

La Figura 4.13 muestra como es realizada la definición de un conector. Este, al igual que los componentes tiene un descripción y un id. Sin embargo, este conector en particular tiene dos interfaces que tiene una dirección de

entrada/salida. Es importante recalcar que a la hora de hacer referencia a un elemento dentro de la configuración, esto se hará mediante la invocación del id y no de la descripción.

```
49  <types:archTypes xsi:type="types:ArchTypes">
50      <types:componentType types:id="componentTypeffa80004-19c35184-3ae83698-3d56056c" xsi:type=
    "types:ComponentType">
51          <types:description xsi:type="instance:Description">numero</types:description>
52      </types:componentType>
53      <types:connectorType types:id="connectorTypeffa80004-19c3d56a-1ba82c43-3d56056d" xsi:type=
    "types:ConnectorType">
54          <types:description xsi:type="instance:Description">red</types:description>
55      </types:connectorType>
56      <types:interfaceType types:id="interfaceTypeffa80004-19c41f73-e120bdda-3d56056e" xsi:type=
    "types:InterfaceType">
57          <types:description xsi:type="instance:Description">puerto1</types:description>
58      </types:interfaceType>
59  </types:archTypes>
```

Figura 4.14: Ejemplo de creación de tipos particulares en XADL2.0

La Figura 4.14, muestra como son definidos tipos particulares de componentes, conectores e interfaces. Si se ve la Figura 4.14 existe un tipo de componente número, su definición dentro de la sintaxis de xADL2.0 se hace en las líneas 50 a 52 de la Figura 4.14. Así podemos ver la definición del tipo de conector red, y el tipo de interfaz puerto1. Todo esto debe estar anidado dentro de la etiqueta ArchTypes, y es externo a la estructura de componentes.

Hasta aquí es posible ver la sintaxis básica de xADL2.0 y su forma de trabajo. A manera de resumen se presenta la siguiente tabla:

Estructura Arquitectural	Sintaxis en xADL2.0	Atributos en xADL2.0
Arquitectura	<instance:xArch … > … </instance:xArch>	Como tal un elemento xArch no tiene atributos sino elementos anidados

Estructura Arquitectural	Sintaxis en xADL2.0	Atributos en xADL2.0
Estructura	<types:archStructure types:id=”…” xsi:type=”types:ArchStructure”> … </types:archStructure>	Tiene un atributo id, que lo identificara en caso de que otro elemento quisiera referirse a la estructura.
Componente	<types:component types:id="…" xsi:type="types:Component"> <types:description xsi:type="instance:Description">…</types:description> … </types:component>	Tiene un id, y un elemento anidado descripción. La descripción podría ser un nombre o algo que mostrara la naturaleza del componente de manera explícita.
Enlace	<types:link types:id="…” xsi:type="types:Link"> <types:description xsi:type="instance:Description">…</types:description> <types:point xsi:type="instance:Point"> <instance:anchorOnInterface xlink:href="Referencia al id del objeto al cual se une el punto" xsi:type="instance:XMLLink" xlink:type="simple"/> </types:point> <types:point xsi:type="instance:Point"> <instance:anchorOnInterface xlink:href="Referencia al id del objeto al cual se une el punto" xsi:type="instance:XMLLink" xlink:type="simple"/> </types:point> </types:link>	Un enlace posee una descripción, y dos elementos anidados, los cuales son los puntos extremos, los cuales a su vez guardan una referencia al id del objeto al cual están unidos.
Interfaz	<types:interface types:id="…" xsi:type="types:Interface"> <types:description xsi:type="instance:Description">…</types:description> <types:direction xsi:type="instance:Direction">…</types:direction> </types:interface>	Las interfaces son los puntos de salida de un componente o conector. Tiene una descripción y una dirección. La dirección varía entre los valores in, out e inout. Una interfaz al pertenecer a un

Estructura Arquitectural	Sintaxis en xADL2.0	Atributos en xADL2.0
		elemento, debe estar encerrada en las etiquetas del objeto al cual pertenece.
Conector	<types:connector types:id="…" xsi:type="types:Connector"> <types:description xsi:type="instance:Description">…</types:description> </connector>	Un conector tiene una descripción y un id.
Tipos particulares.	<types:componentType types:id="…" xsi:type="types:ComponentType"> <types:description xsi:type="instance:Description">…</types:description> </types:componentType>	Un tipo nuevo debe tener una descripción y un id. Podemos definir: ComponentType ConnectorType InterfaceType

Tabla 4.5: Resumen sintaxis básica xADL2.0

Como complemento a lo anterior, hay que aclarar que xADL2.0 maneja diferentes tipos de esquemas XML para las diferentes etapas en la especificación de una arquitectura. El manejo de estos esquemas se puede ver en la siguiente tabla [Dashofy, 2007]:

Instancia (Tiempo de ejecución)	Estructura (Tiempo Diseño)	Tipo (Tiempo Diseño)
Instancia de Componente	Componente	Tipo componente
Instancia de Conector	Conector	Tipo conector
Instancia de interface	Interface	Tipo Interface.
Instancia de Enlace	Enlace	--
Grupo	Grupo	--

Tabla 4.6: Relación entre tipos, estructura e instancias en xADL2.0

Para el caso del proyecto planteado en este documento, se manejaran los esquemas relacionados únicamente con el tiempo de diseño [ISR, 2007].

5 MODELO FUNCIONAL

A continuación se presenta el modelo funcional para el desarrollo de una aplicación que permita traducir descripciones de arquitecturas de software basadas en la sintaxis de xADL 2.0 extendido a especificaciones formales basadas en la sintaxis del cálculo para el modelamiento formal de arquitecturas de software – ρ_{arq}.

Este trabajo se ha desarrollado por medio de la identificación de requerimientos, tanto funcionales como no funcionales, que han marcado las pautas a seguir y ha permitido estructurar las funcionalidades de este software utilizando los casos de uso.

Ciñéndose a la estructura que propone la metodología RUP [Kruchten, 2001] (Rational Unified Process), en este documento, que es producto de la fase de inicio, se establecen ámbitos y límites del proyecto, se presenta una visión general de los requerimientos del proyecto, características clave, riesgos y restricciones principales, así como se determina un glosario inicial que incluye terminología clave del dominio y finalmente se obtiene un modelo inicial de casos de uso con su respectiva especificación.

5.1 PROPÓSITO

Describir las funcionalidades de un prototipo de aplicación que permita realizar la traducción de descripciones de arquitecturas de software realizadas en xADL2.0 extendido al cálculo de especificación formal – ρ_{arq}, así como la interacción entre un usuario y el mismo software.

5.2 DESCRIPCIÓN DEL PROBLEMA

5.2.1 Dominio del problema

La continua evolución en el proceso de construcción de software ha generado que el modelamiento de arquitecturas sea un tema de constante investigación para el análisis y la comprensión global de un sistema a nivel estático y de comportamiento. Como resultado significativo han surgido los Lenguajes de Descripción Arquitectural – LDA (del inglés Architecture Description Languages ADLs), los cuales han posibilitado representar una arquitectura de software en términos de sus componentes, conectores y las diferentes configuraciones que pueden surgir entre esos elementos. Además de los LDA's, se han creado métodos de especificación formal de software, que involucran técnicas matemáticas para describir el comportamiento de un sistema. Gracias a que estos métodos usan álgebras de procesos en su mayoría, se puede afirmar que reducen la ambigüedad que presentan en algunas ocasiones los LDA's y otros métodos convencionales u orientados a objetos [Pressman, 2002].

Entre los muchos LDA's existentes, se destaca xADL2.0 porque fue construido con el objetivo de ser altamente extendible, razón por la cual está basado en XML. Adicionalmente, es un lenguaje modular que permite el modelamiento de diferentes estructuras arquitecturales y la extensión de las mismas.

Complementariamente, el cálculo – ρ_{arq} es un método de especificación formal [para arquitecturas de software basadas en componentes] [ACSC, 1995] cuya idea subyacente es que a partir de una configuración estática

inicial se pueda pasar a una nueva configuración por medio de la aplicación de las reglas de la semántica operacional del mismo. [Diosa et al., 2005].

Sin embargo, aún no se ha planteado la posibilidad de aprovechar la extendibilidad que ofrece xADL 2.0 como un lenguaje de descripción estática de arquitecturas junto con las ventajas que ofrece el cálculo para modelamiento formal – ρ_{arq}. Es por esto que se plantea la elaboración de un mecanismo que sirva de puente entre estos dos métodos de descripción arquitectural, aprovechando las cualidades de ambos como aporte en el modelamiento formal de arquitecturas de software.

El mecanismo propuesto, permite traducir una configuración estática descrita con xADL 2.0 a una configuración basada en un método de especificación formal, para luego describir el comportamiento de sistemas constituidos por componentes usando la semántica operacional del cálculo –ρ_{arq}.

5.2.2 Restricciones

El software a desarrollar en este proyecto posee tres restricciones básicas. La primera se refiere a que no estará orientado a la web y por consiguiente el trabajo con el mismo será únicamente local. La segunda restricción se refiere a que será desarrollado para ser utilizado únicamente por un usuario a la vez. Por último, la detección de errores de sintaxis se hará únicamente durante el proceso de traducción más no en tiempo de escritura del código a traducir y únicamente se realizará para el archivo de entrada XML.

5.2.3 Características del usuario

Para el uso de esta herramienta como tal, es necesario poseer conocimientos básicos en descripción de arquitecturas de software, ya sea con el uso de xADL2.0 o con el cálculo formal – ρ_{arq}.

La herramienta deberá poseer una interfaz grafica amigable, que le permita al usuario navegar a través de ella de manera intuitiva, teniendo en cuenta los mínimos conocimientos previos acerca del proceso que se llevará a cabo con ayuda de la misma.

5.3 REQUERIMIENTOS INICIALES

En esta sección, se identifican los requerimientos para el desarrollo del software planteado anteriormente. En primer lugar se hará una descripción general (sección 5.3.1) de lo que debe hacer el software para luego elaborar una especificación individual de los requerimientos, dividiéndolos en funcionales y en no funcionales (Secciones 5.3.2 y 5.3.3).

5.3.1 Definición de requerimientos

De acuerdo a la información obtenida del Profesor Henry Alberto Diosa se hace a continuación una descripción de los requerimientos para la construcción de un traductor sintáctico entre xADL2.0 y el cálculo formal – ρ_{arq}:

- El software debe permitir importar archivos XML válidos respecto a xADL 2.0 extendido, donde las extensiones de xADL 2.0 tienen el

propósito de soportar conceptos del cálculo formal – ρ_{arq}[Arquisoft, 2008].

- El software debe poseer un mecanismo de validación de la sintaxis de xADL2.0 para el archivo XML de entrada, en tiempo de lectura de dicho archivo.

- El software debe permitir la generación de un archivo de texto que sea conforme a la sintaxis del cálculo formal – ρ_{arq} para la especificación de arquitecturas de software.

- El proceso de traducción se hará utilizando un motor de transformación que analice un archivo XML y obtenga un documento de texto.

- El software debe permitir guardar el archivo de texto generado con el código basado en el cálculo – ρ_{arq}.

5.3.2 Requerimientos funcionales

Para efectos del presente documento, se coloca RF y el número del requerimiento referenciado:

REFERENCIA	REQUERIMIENTO
RF – 1	Importar archivo XML
RF – 1.1	Verificar que el archivo de entrada al aplicativo sea conforme XML y válido respecto de xADL 2.0 extendido

REFERENCIA	REQUERIMIENTO
RF – 1.1.1	Cargar el esquema extendido de xADL 2.0 para la validación del archivo XML de entrada
RF – 2	Transformar el contenido del archivo XML importado a un código conforme a la sintaxis del cálculo – ρ_{arq}
RF – 2.1	Ubicar la hoja de estilos XSL para la comparación con el archivo de entrada XML
RF – 2.2	Mostrar la transformación del archivo XML importado al código conforme a la sintaxis del cálculo – ρ_{arq}
RF – 3	Generar documento en código LaTex con la transformación conforme a la sintaxis del cálculo – ρ_{arq}
RF – 4	Guardar el documento de texto en código LaTex, con la transformación conforme a la sintaxis del cálculo – ρ_{arq}

Tabla 5.1: Requerimientos funcionales

5.3.3 Requerimientos no funcionales

A continuación se presentan los requerimientos no funcionales identificados para el caso de estudio. Para efectos que en el presente documento, se quiera referenciar alguno de los requerimientos no funcionales, colocaremos RNF y el número del requerimiento referenciado:

REFERENCIA	ATRIBUTOS	REQUERIMIENTOS
RNF-1	Usabilidad	• Gráficamente atractivo (manejo por botones intuitivamente ubicados). • Navegación rápida a través de atajos de teclado. • Intuitivo para cualquier usuario.

REFERENCIA	ATRIBUTOS	REQUERIMIENTOS
RNF-2	Fiabilidad	• Manejo de excepciones para la tolerancia a fallas en la ubicación de archivos locales.
RNF-3	Portabilidad	• El software debe funcionar independientemente del sistema operativo en el cual se encuentra.
RNF-4	Entrega	• Se entregará el modelo del sistema y el software desarrollado, de acuerdo con la estimación respecto a tiempo de entrega, presupuesto, esfuerzo, etc.
RNF-5	Implementación	• Lenguaje Java SDK 1.6 • Sistemas Operativos: Multiplataforma • Soporte para un único idioma español.
RNF-6	Documentación	• Se entregará un manual de usuario, con información operacional que describe el uso del software considerando: opciones de configuración, opciones de su uso. Además debe permitir ser imprimible.

Tabla 5.2: Requerimientos no funcionales

5.4 MODELO FUNCIONAL BASADO EN CASOS DE USO

En esta sección, crearemos los casos de uso necesarios, para cubrir los requerimientos de la aplicación, identificados anteriormente.

5.4.1 Diagrama general de casos de uso

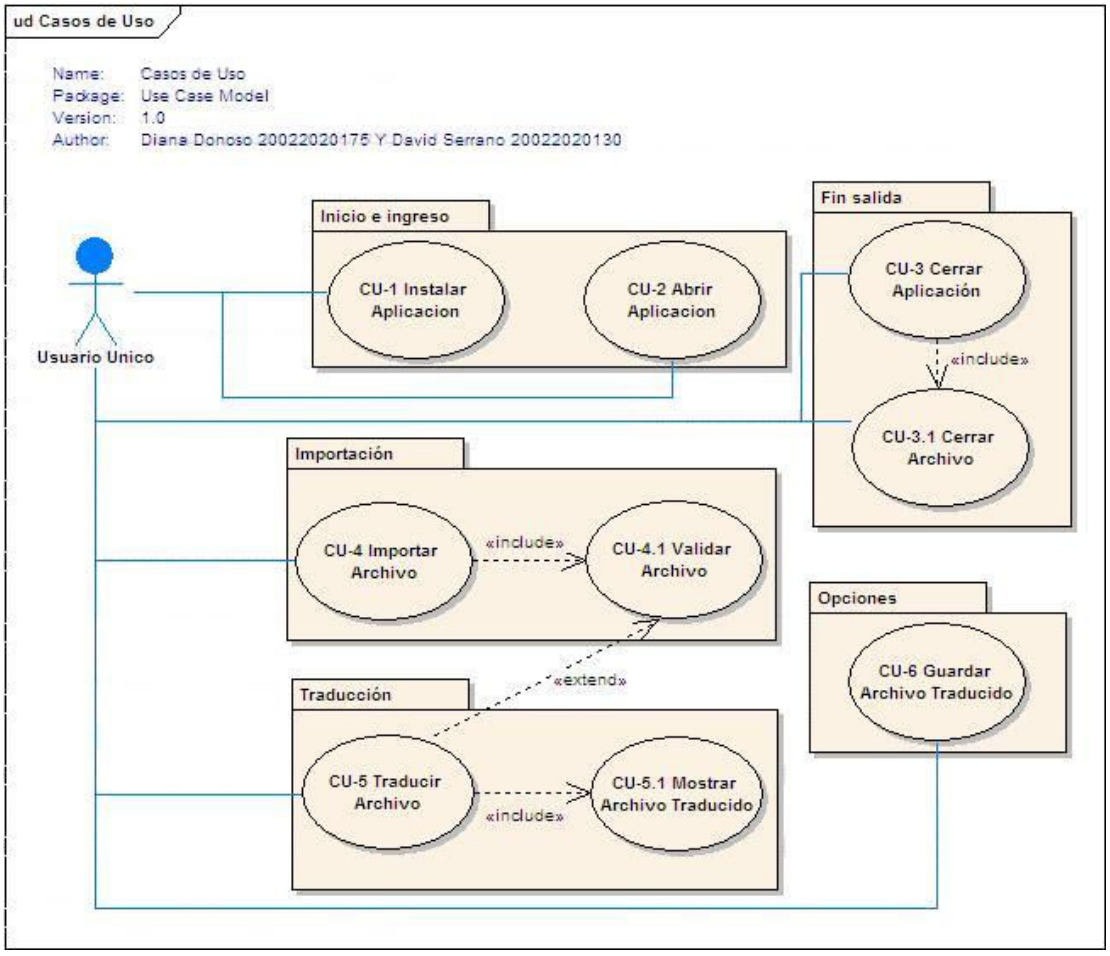

Figura 5.1: Diagrama general de casos de uso

5.4.2 Referencia cruzada de caso de uso vs. Requerimientos

En esta sección, crearemos los casos de uso necesarios, para cubrir los requerimientos de la aplicación, identificados anteriormente.

CASO DE USO	NOMBRE	REQUERIMIENTO
CU-1	Instalar aplicación	RF–1, RF–2, RF–3, RF–4
CU-2	Abrir aplicación	RF–1, RF–2, RF–3, RF–4

CASO DE USO	NOMBRE	REQUERIMIENTO
CU-3	Cerrar aplicación	RF–4
CU-3.1	Cerrar archivo	RF–4
CU-4	Importar archivo XML	RF–1
CU-4.1	Validar archivo XML	RF–1, RF–1.1
CU-5	Transformar archivo	RF–2, RF–2.1, RF–2.2, RF–3
CU-5.1	Mostrar archivo transformado	RF–2.2
CU-6	Guardar archivo	RF–3, RF–4

Tabla 5.3: Casos de uso

5.4.3 Especificación de casos de uso en formato expandido

A continuación se presentan los casos de uso identificados en formato expandido, esto con el fin de hacer el sistema más comprensible en cuanto a la secuencia de actividades necesarias para cumplir cada caso de uso señalado.

5.4.3.1 Subsistema inicio e ingreso

Campo	*Valor*
ID	CU-1
Caso de uso	Instalar Aplicación
Actores	Usuario Único
Descripción	Muestra las diferentes opciones de la aplicación para colocar dentro del sistema, de forma que esta funcione sin complicaciones.
Tipo	Primario
Precondiciones	-
Post-condiciones	El sistema muestra la pantalla de progreso de la instalación durante todo el proceso e instala en disco

Campo	*Valor*
ID	CU-1
	duro las librerías requeridas, en tiempo de ejecución, de acuerdo con la selección del usuario.
RC Casos de uso	-
RC Requerimientos	RF–1, RF–2, RF–3, RF–4

Escenario Normal	
Acción actores	*Respuesta sistema*
1. El usuario inicia la instalación.	
	2. Muestra un mensaje de bienvenida de la instalación con la descripción del software y las utilidades, con dos botones **Siguiente** y **Cancelar**.
3. Selecciona la opción **Siguiente.**	
	4. La aplicación muestra un campo para la ruta de ubicación de los archivos y carpetas necesarias para su manejo con un botón **Examinar** para seleccionar tal ubicación. Adicionalmente muestra tres botones **Atrás**, **Siguiente**, **Cancelar**.
5. El usuario selecciona la ubicación dando click en la opción **Examinar,** eligiendo la ruta deseada para la instalación y elige **Siguiente**.	
	6. Crea las carpetas y archivos de la aplicación en la ubicación seleccionada por el usuario.
	7. Termina la instalación mostrando mensaje de éxito.
8. El usuario acepta la finalización.	

Campo	*Valor*
ID	CU-1
Escenario Alternativo	
En los paso 3 o 5 del escenario normal, el usuario selecciona el botón **Cancelar** la instalación.	1. El sistema muestra un mensaje de confirmación con el texto "¿Está seguro que quiere abortar la instalación?", con dos botones **Sí** y **No**.
2. El usuario selecciona el botón **Sí**.	3. Cancela todos los procesos involucrados con la instalación y cierra el asistente para instalación.
En el paso 5 del escenario normal, el usuario selecciona la opción **Atrás.**	4. Regresa al paso 2 del escenario normal en el CU-1.
En el paso 2 del escenario alternativo, el usuario selección la opción **No**.	5. Continúa la operación que estaba realizando antes que el usuario seleccionara la opción **Cancelar**.
Escenario Excepcional	
En el paso 5 del escenario normal, el usuario no especifica ubicación, deja el campo vacío o ingresa valores incorrectos.	1. Muestra un mensaje de alerta, solicitando completar o cambiar los campos incompletos o incorrectos, según sea el caso y retorna al paso 4 del escenario normal en el CU-1.

Tabla 5.4: CU-1 Instalar aplicación

Campo	*Valor*
ID	CU-2
Caso de uso	Abrir Aplicación
Actores	Usuario Único
Descripción	Este proceso pone en funcionamiento la aplicación para poder trabajar con ella
Tipo	Primario
Precondiciones	Haber realizado el CU-1
Post-condiciones	La aplicación queda lista para ser usada.
RC Casos de uso	--

Campo	*Valor*	
ID	CU-2	
RC Requerimientos	RF–1, RF–2, RF–3, RF–4	
Escenario Normal		
Acción actores		***Respuesta sistema***
1. El usuario ejecuta la aplicación.		
		2. El sistema ubica y carga todas las librerías y procesos necesarios para iniciar la aplicación.
		3. Muestra un mensaje de bienvenida y habilita todas las opciones disponibles para su uso.
Escenario Alternativo		
Escenario Excepcional		
En el paso 2 del escenario normal, el sistema no encuentra alguna de las librerías o no puede cargar alguno de los procesos necesarios para iniciar la aplicación.		1. Muestra un mensaje de error informando que es imposible ejecutar la aplicación, con un botón **Aceptar.**

Tabla 5.5: CU-2 Abrir aplicación

5.4.3.2 Subsistema fin salida

Campo	*Valor*
ID	CU-3
Caso de uso	Cerrar Aplicación
Actores	Usuario único
Descripción	Cierra el aplicativo, deteniendo todos los procesos relacionados con esta y que se estén ejecutando en ese momento.
Tipo	Primario
Precondiciones	Haber realizado el CU-2
Post-condiciones	El aplicativo se cierra, cerrando los archivos locales relacionados con el mismo.

Campo	***Valor***
ID	CU-3
RC Casos de uso	--
RC Requerimientos	RF–4
Escenario Normal	
Acción actores	***Respuesta sistema***
1. Selecciona cerrar aplicación.	
	2. Muestra un cuadro de verificación preguntando si está seguro de cerrar la aplicación, con dos botones **Aceptar** y **Cancelar**.
3. Elige aceptar.	
	4. Verifica que no haya ningún archivo abierto y sin guardar.
	5. Finaliza todos los procesos que se estén ejecutando en ese momento, relacionados con la aplicación y la cierra.
Escenario Alternativo	
En el paso 3 del escenario normal, elige cancelar	1. Regresa a la última pantalla de la aplicación en la que se estaba trabajando.
Escenario Alternativo	
En el paso 4 del escenario normal, la verificación determina que si hay uno o más archivos abiertos sin guardar.	1. Invoca el paso 2 del CU-6 por cada uno de los archivos abiertos.

Tabla 5.6: CU-3 Cerrar aplicación

Campo	***Valor***
ID	CU-3.1
Caso de uso	Cerrar Archivo
Actores	Usuario único
Descripción	Cierra el archivo sobre el que se está trabajando.
Tipo	Primario
Precondiciones	Haber realizado el CU-4

Campo	***Valor***
ID	CU-3.1
Post-condiciones	El archivo se cierra, dejando abiertos los demás archivos abiertos simultáneamente.
RC Casos de uso	--
RC Requerimientos	RF–4
Escenario Normal	
Acción actores	***Respuesta sistema***
1. Elige cerrar archivo.	
	2. Muestra un mensaje de confirmación de cierre del archivo con botones **Aceptar** y **Cancelar**
3. Elige la opción **Aceptar**.	
	4. Verifica que no haya ningún archivo traducido abierto y sin guardar.
	5. Cierra el archivo y retorna a la pantalla principal de la aplicación.
Escenario Alternativo	
En el paso 3 del escenario normal, elige cancelar	1. Regresa a la última pantalla de la aplicación en la que se estaba trabajando.
En el paso 2 del escenario excepcional, el usuario selecciona **Cancelar**.	2. Regresa al paso 5 del escenario normal del CU-3.1
Escenario Excepcional	
En el paso 4 del escenario normal, la verificación determina que si hay uno o más archivos abiertos sin guardar.	1. Por cada uno de los archivos abiertos sin guardar, muestra un mensaje de confirmación preguntando si desea guardar el archivo, con dos botones **Aceptar** y **Cancelar**.
2. Selecciona la opción **Aceptar**.	3. Invoca el paso 2 del CU-6 por cada uno de los archivos abiertos sin guardar.

Tabla 5.7: CU-3.1 Cerrar aplicación

5.4.3.3 Subsistema importación

Campo	*Valor*
ID	CU-4
Caso de uso	Importar Archivo
Actores	Usuario Único
Descripción	Carga un documento XML para ser traducido, previa validación contra el esquema de xADL 2.0 extendido
Tipo	Primario
Precondiciones	Haber realizado el CU-2
Post-condiciones	Carga un documento XML válido con respecto al esquema de xADL 2.0 extendido para ser traducido
RC Casos de uso	CU-4.1
RC Requerimientos	RF–1

Escenario Normal	
Acción actores	***Respuesta sistema***
1. Selecciona la opción importar archivo XML	
	2. Muestra una ventana solicitando la ruta de ubicación del archivo XML que desea importar con un botón **Examinar** para seleccionar tal ubicación. Adicionalmente muestra dos botones **Aceptar** y **Cancelar**.
3. El usuario selecciona la ubicación dando click en la opción **Examinar,** eligiendo la ruta deseada para la instalación y elige **Aceptar**.	
	4. Verifica que la extensión del archivo sea *.xml
	5. Invoca el CU-4.1

Campo	*Valor*
ID	CU-4
Escenario Alternativo	
En el paso 3 del escenario normal, el usuario elige cancelar	1. Regresa a la última pantalla de la aplicación en la que se estaba trabajando.
Escenario Excepcional	
En el paso 3 del escenario normal, el usuario no especifica ubicación, deja el campo vacío o ingresa valores incorrectos.	1. Muestra un mensaje de alerta, solicitando completar o cambiar los campos incompletos o incorrectos, según sea el caso y retorna al paso 2 del escenario normal en el CU-4.
En el paso 4 del escenario normal, la extensión del archivo seleccionado por el usuario es diferente a *.xml	Muestra mensaje de error indicando que no se puede importar el tipo de archivo seleccionado

Tabla 5.8: CU-4 Importar archivo

Campo	*Valor*
ID	CU-4.1
Caso de uso	Validar archivo XML
Actores	Usuario Único
Descripción	Carga un documento XML para ser validado
Tipo	Primario
Precondiciones	Haber realizado el CU-2
Post-condiciones	Muestra los resultados de la validación de la sintaxis del documento XML contra el esquema de xADL 2.0 extendido
RC Casos de uso	CU-4
RC Requerimientos	RF–1, RF–1.1
Escenario Normal	
Acción actores	***Respuesta sistema***
1. Viene del paso 5 del CU-4.	
	2. Carga en memoria el documento XML a importar.

Campo	*Valor*	
ID	CU-4.1	
		3. Ubica el archivo que contiene el esquema contra el cual se hará la validación del documento XML a importar.
		4. Valida el documento XML seleccionado por el usuario contra el esquema de xADL2.0 extendido.
		5. La validación es exitosa; muestra un mensaje de éxito de la validación del documento XML con un botón **Aceptar**.
6. Elige **Aceptar**		
		7. Muestra el contenido del archivo XML importado y válido contra el esquema de xADL 2.0 extendido, en el costado izquierdo de la ventana dividida.
Escenario Alternativo		
En el paso 5 del escenario normal, la validación no tiene éxito.		1. Muestra mensaje de error de la validación con un botón **Aceptar**, no carga el archivo XML y regresa a la pantalla principal de la aplicación.
Escenario Excepcional		
En el paso 3 del escenario normal, el sistema no encuentra el archivo que contiene el esquema contra el cual se hará la validación del documento XML a importar.		1. Muestra mensaje de error en la ubicación del esquema para la validación con un botón **Aceptar** y regresa a la pantalla principal de la aplicación.

Tabla 5.9: CU-4.1 Validar archivo XML

5.4.3.4 Subsistema traducción

Campo	*Valor*
ID	CU-5
Caso de uso	Traducir Archivo
Actores	Usuario Único
Descripción	Traduce el contenido de un archivo XML previamente importado a un nuevo documento de texto en código LaTex conforme a la sintaxis del cálculo formal $-\rho_{arq}$.
Tipo	Primario
Precondiciones	Haber realizado previamente el CU-4 Importar Archivo.
Post-condiciones	Crea internamente un documento de texto con la transformación del archivo XML de entrada con la sintaxis del cálculo formal $-\rho_{arq}$.
RC Casos de uso	CU-5.1
RC Requerimientos	RF–2, RF–2.1, RF–2.2, RF–3
Escenario Normal	
Acción actores	***Respuesta sistema***
1. Selecciona traducir archivo.	
	2. Ubica la hoja de estilo XSLT para iniciar la transformación del documento XML importado.
	3. Toma el documento XML importado y previamente validado e inicia la transformación.
	4. Genera un archivo de extensión *.tex en memoria, con la transformación terminada.
	5. Genera un archivo de extensión *.png en memoria,

Campo	*Valor*
ID	CU-5
	con la visualización de la expresión resultante de la transformación respectiva.
	6. Muestra un mensaje informando que la traducción fue terminada con éxito, con un botón **Aceptar**.
7. Selecciona **Aceptar**.	
	8. Invoca al caso de uso CU-5.1.
Escenario Alternativo	
Escenario Excepcional	
En el paso 2 del escenario normal, no se encuentra la hoja de estilos para la transformación.	1. Genera un mensaje de error indicando que no se encontró la hoja de estilos para la transformación, sugiriendo la reinstalación de la aplicación, con un botón **Aceptar**.
En el paso 3 del escenario normal, no se puede iniciar o hay errores en la transformación del documento XML importado.	2. Genera un mensaje de error indicando que hubo errores durante la transformación, sugiriendo la revisión de la estructura del documento XML importado, con un botón **Aceptar**.

Tabla 5.10: CU-5 Transformar archivo

Campo	*Valor*
ID	CU-5.1
Caso de uso	Mostrar Archivo Traducido
Actores	Usuario Único
Descripción	Muestra el archivo interno traducido previamente, en la ventana principal del aplicativo.

Campo	*Valor*
ID	CU-5.1
Tipo	Primario
Precondiciones	Haber realizado el CU-5
Post-condiciones	La aplicación queda en la ventana principal con el archivo traducido activo.
RC Casos de uso	CU-5
RC Requerimientos	RF–2.2
Escenario Normal	
Acción actores	***Respuesta sistema***
1. Viene del paso 7 del CU-5	
	2. Limpia el área reservada para mostrar los archivos traducidos en la ventana dividida de la aplicación.
	3. Toma los archivos de extensión *.tex y *.png guardados en memoria luego del proceso de transformación y visualiza tanto la interpretación en forma de imagen de la expresión resultante, como su contenido, en el área reservada para esta función.
Escenario Alternativo	
Escenario Excepcional	

Tabla 5.11: CU-5.1 Mostrar archivo traducido

5.4.3.5 Subsistema opciones

Campo	*Valor*
ID	CU-6
Caso de uso	Guardar Archivo

Campo	*Valor*
ID	CU-6
Actores	Usuario Único.
Descripción	Guarda un documento de extensión *.tex con la transformación del contenido del documento XML importado a un nuevo documento de texto en código LaTex conforme a la sintaxis del cálculo formal $-\rho_{arq}$ y un archivo de extensión *.png con la imagen de la interpretación del código LaTex de tal transformación.
Tipo	Primario
Precondiciones	Haber realizado el CU-5
Post-condiciones	Crea dos archivos nuevos de extensión *.tex y *.png en el disco duro del computador del usuario, en la ubicación que este decida.
RC Casos de uso	--
RC Requerimientos	RF–3, RF–4

Escenario Normal	
Acción actores	*Respuesta sistema*
1. Selecciona guardar archivo o viene del paso 1 del escenario excepcional del CU-3.1	
	2. Muestra una nueva ventana solicitando nombre del archivo y ubicación del mismo, con dos botones **Aceptar** y **Cancelar**.
3. Selecciona la ruta deseada para la ubicación del archivo a guardar, indica el nombre que le desea asignar y elige **Aceptar**.	
	4. Verifica que no haya archivos de extensiones *.tex y/o *.png en la misma ruta y con el mismo nombre seleccionados por el usuario.

Campo	*Valor*	
ID	CU-6	
		5. Crea en el disco duro un archivo de extensión *.tex y otro de extensión *.png, en la ruta y con el nombre especificados por el usuario, con el contenido de la transformación producida previamente y la imagen de la expresión resultante.
		6. Retorna a la pantalla principal de la aplicación o si viene del paso 1 del escenario excepcional del CU-3.1, sigue al paso 5 del escenario normal de ese mismo caso de uso.
Escenario Alternativo		
En el paso 3 del escenario normal, elije la opción **Cancelar**.		1. Se cierra el cuadro de dialogo de solicitud de datos y se continua en el paso 5 del escenario normal de este mismo caso de uso.
En el paso 2 del escenario excepcional, el usuario selecciona **Cancelar.**		1. Retorna al paso 2 del escenario normal del CU-6.
Escenario Excepcional		
En el paso 4 del escenario normal, existe un archivo de extensión *.tex y/o *.png en la misma ruta y con el mismo nombre seleccionados por el usuario.		1. Muestra un mensaje de alerta informando que ya existe un archivo con el mismo nombre seleccionado y pregunta si desea sobrescribirlo, con dos botones **Aceptar** y **Cancelar.**
2. El usuario selecciona **Aceptar**.		3. Borra el archivo existente con el mismo nombre y en la ubicación seleccionados por el usuario y guarda los archivos que contienen la transformación producida

Campo	*Valor*	
ID	CU-6	
		previamente y la imagen de la expresión resultante con el nombre y en la ubicación indicados.
En el paso 5 del escenario normal, no hay espacio en disco.		1. Muestra un mensaje de error indicando que no hay espacio suficiente en disco para guardar los archivos con la transformación hecha, no guarda ningún archivo y regresa a la última pantalla de la aplicación en la que se estaba trabajando.

Tabla 5.12: CU-6 Guardar archivo traducido

6 DISEÑO DE LA ARQUITECTURA

En esta sección se presenta el diseño de la arquitectura para el desarrollo del software propuesto en el proyecto, identificando estilos y patrones arquitecturales adecuados para este tipo de aplicación.

6.1 ESTILOS ARQUITECTURALES IDENTIFICADOS

En esta sección buscaremos definir los estilos arquitecturales adecuados para un sistema de traducción sintáctica entre el lenguaje de descripción arquitectural xADL 2.0 extendido y el cálculo para el modelamiento formal de arquitecturas de software – ρ_{arq}, teniendo en cuenta que serán estos estilos los que identifiquen la aplicación frente a otras de su tipo. Además servirá también para mejorar y esclarecer el proceso de desarrollo. Los estilos que se

tuvieron en cuenta para esta sección, pueden consultarse en [Pressman, 2002]:

- Arquitectura de flujo de datos ó de canal-filtro (pipes and filters). Se propone utilizar este estilo, porque se va a realizar un proceso de transformación de datos de entrada a través de una serie de módulos de software que actúan como una serie de filtros y generan ciertos datos de salida, como se muestra en la siguiente gráfica.

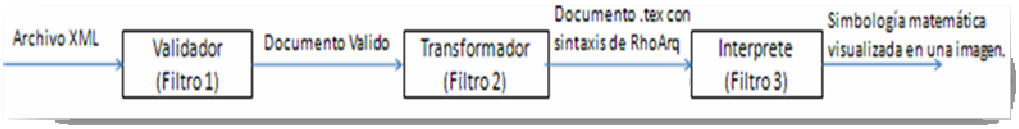

Figura 6.1: Representación de la Arquitectura de Flujo de Datos

Dentro de esta arquitectura existen dos tipos de transformaciones, donde en la gráfica anterior muestra una transformación secuencial por lotes, dado que el paso de datos entre los filtros se hace de manera secuencial.

- Arquitectura de implementación estratificada. Se justifica el uso de este estilo, dado que la aplicación estará dividida en las siguientes capas:

    - Presentación. Contiene todos los objetos y métodos necesarios para la interacción del usuario con la aplicación.

    - Modelo de Dominio. Contiene todos los objetos y métodos necesarios para los procesos de validación y traducción sintáctica entre el lenguaje de descripción arquitectural xADL 2.0

extendido y el cálculo para el modelamiento formal de arquitecturas de software – ρ_{arq}.

- Persistencia. Contiene los datos externos necesarios para ejecutar los procesos de validación y traducción sintáctica entre el lenguaje de descripción arquitectural xADL 2.0 extendido y el cálculo para el modelamiento formal de arquitecturas de software – ρ_{arq}. Como datos persistentes clasifican:

  - Esquemas XML: Estos archivos son necesarios para la validación de un documento XML y corresponden a los esquemas propios de xADL2.0 junto con dos esquemas de extensión de este lenguaje, los cuales sirven para dar soporte a conceptos propios del cálculo para el modelamiento formal de arquitecturas de software – ρ_{arq}.

  - Hojas de estilo XSL: Estos archivos son necesarios para la transformación de código XML a código Latex.

6.2 PATRONES ARQUITECTURALES IDENTIFICADOS

Anteriormente se propusieron algunos estilos arquitecturales que identificarían la aplicación, ahora debemos definir aquellos patrones arquitecturales que nos permitan cumplir con los estilos propuestos:

- Capas. Ya anteriormente se mencionó que se usará el estilo arquitectural de estratificación. Esto nos obliga a implementar dentro de la aplicación el patrón arquitectural por capas. A continuación mostraremos un esquema que podría explicar esto con mayor claridad:

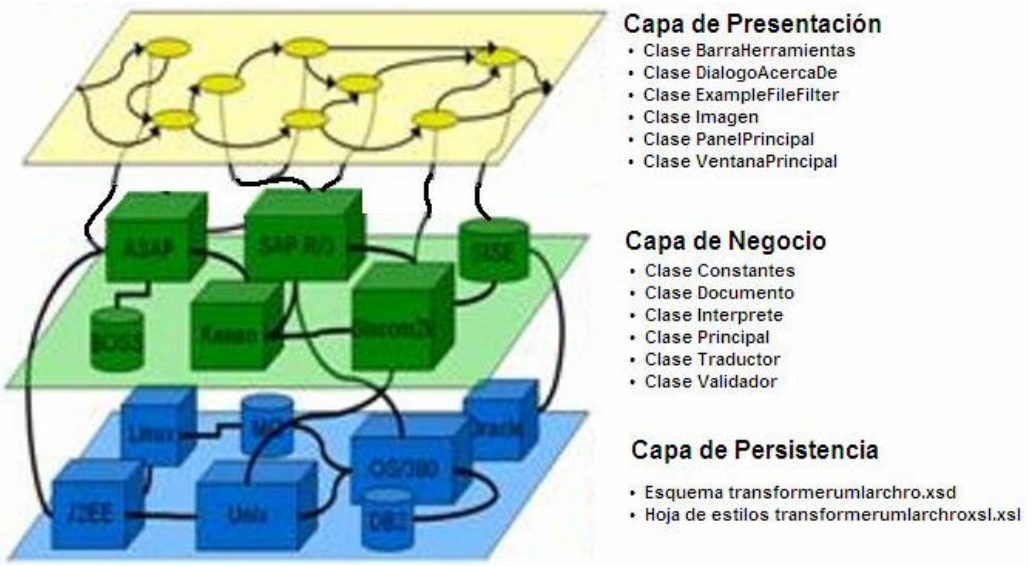

Figura 6.2: Implementación General del Patrón Arquitectural por Capas.

Se puede observar la comunicación entre los diferentes componentes de cada capa, que en conjunto permiten que las funcionalidades propuestas para el proyecto sean llevadas cabo.

- Tuberías y filtros. Dado que la aplicación maneja un flujo de datos claro, el patrón arquitectural tuberías y filtros, es el indicado para administrarlo. De manera general, este patrón propone la división del problema general en diferentes sub-tareas que aunque son realizadas de manera independiente son complementarias entre sí, para resolver el problema general inicialmente planteado. En el caso de este proyecto, se plantean dos sub-tareas: validación y traducción. El formato de los datos que pasan por la primera etapa está definido como un archivo de entrada XML, mientras que en la etapa de traducción se toma un archivo XML validado. Nuestro repositorio de datos está definido como un archivo de salida de texto plano con sintaxis del cálculo formal – ρ_{arq}. La figura 6.2 muestra de manera explícita lo anteriormente expuesto.

- Invocación explicita. Este patrón propone que las solicitudes de servicios por parte de objetos cliente a otros objetos dentro de la aplicación, se haga de manera explícita, enviándole los datos necesarios y suficientes para que el objeto servidor pueda suplir los servicios solicitados por el cliente.

  Un ejemplo de esto, dentro del proyecto es el proceso de validación. Allí, es necesario enviar datos como la ubicación del archivo, la extensión y el código contenido en el archivo, para que el objeto "validador" logre ejecutar el servicio respectivo. Este proceso se inicia con el evento explícito de solicitud del usuario a través de un objeto gráfico que captura los datos antes mencionados.

7 MODELO ESTRUCTURAL

7.1 PROPÓSITO

Identificar las clases necesarias para la construcción de un traductor sintáctico entre el lenguaje de descripción arquitectural xADL 2.0 extendido y el cálculo para el modelamiento formal de arquitecturas de software – ρ_{arq}. La unión coherente de las clases identificadas deberá permitir la implementación completa de las funcionalidades propuestas en el modelo funcional de la aplicación. Dichas clases solo deben relacionarse con el dominio del problema más no con la interfaz grafica u otros aspectos de implementación.

7.2 CLASES CANDIDATAS

A continuación, basados en los casos de uso realizados en el modelo funcional, se proponen las clases que se manejarán dentro del proceso de desarrollo, las cuales tendrán una referencia inicial y una breve descripción.

REFERENCIA	NOMBRE	DESCRIPCIÓN
CC-1	Documento	Representa el documento que se desea traducir, con atributos que describen su contenido, estado, ubicación y otras características para su manejo.
CC-2	Validador	Como su nombre lo indica, valida el documento XML de entrada al traductor contra los esquemas respectivos con el fin de comprobar que esté bien formado y sea válido, para permitir la importación y su respectivo manejo,
CC-3	Constantes	Contiene los valores que permanecen constantes en la aplicación.
CC-4	Intérprete	Encargada de comprender el contenido del documento traducido, el cual se encuentra en código LaTeX, para mostrarlo como símbolos matemáticos y/o su respectiva interpretación.
CC-5	Traductor	Una vez superadas las etapas de validación e importación, esta clase se encarga de convertir el contenido del documento en código LaTeX.
CC-6	TeXFormula	Representa una formula lógica matemática que puede ser visualizada (por la creación de un TeXIcon de este y pintándolo) usando algoritmos basados en los de TeX.

REFERENCIA	NOMBRE	DESCRIPCIÓN
CC-7	File	Las interfaces de usuario y los sistemas operativos usan cadenas de *pathnames* dependientes del sistema a nombres de archivos y directorios. Esta clase presenta una vista, independiente del sistema, de los pathnames jerárquicos.
CC-8	TransformerFactory	Una instancia de *TransformerFactory* puede ser usada para crear objetos transformadores (*Transformer*) y plantillas (*Templates*)
CC-9	SchemaFactory	Esta clase es un compilador de esquemas. Lee representaciones externas de esquemas y los prepara para la validación.
CC-10	Source	Un objeto que implementa esta interface contiene la información necesaria para actuar como un código fuente de entrada (código fuente XML o instrucciones de transformación).
CC-11	Validator	Procesador que verifica un documento XML contra un esquema dado.
CC-12	StringWriter	Una fuente de caracteres que recopila sus resultados en una cadena buffer, la cual puede ser usada para construir una cadena de caracteres
CC-13	StringReader	Una secuencia de caracteres cuya fuente es una cadena de caracteres.
CC-14	Transformer	Una instancia de esta clase abstracta puede transformar un código fuente en otro resultado.

REFERENCIA	NOMBRE	DESCRIPCIÓN
CC-15	StreamSource	Actúa como el propietario de la transformación del código fuente en la forma de una secuencia de marcado XML.
CC-16	Result	Un objeto que implemente esta interface contiene información necesaria para construir el resultado de una transformación.
CC-17	Schema	Este objeto representa un conjunto de restricciones que pueden ser comprobadas y/o ejecutadas contra un documento XML.

Tabla 7.1: Listado de clases candidatas

7.2.1 Diagrama de clases

Con el objetivo de ampliar y aclarar el diseño de la aplicación, se ha creado un diagrama de clases, que especifica las abstracciones más importantes de la lógica ó dominio del negocio. Esto con el fin de cubrir las funcionalidades de la aplicación con formalidad, utilizando el lenguaje unificado de modelado UML 2.0.

7.2.2 Diccionario de clases

A continuación, basados en las clases candidatas identificadas en el comienzo de este documento, se elaborará un diccionario de clases que permitirá comprender mejor las características de cada una de estas clases, pues se explicará su semántica, su tipo, valores por defecto, visibilidad y tipo de retorno, para sus atributos y para sus métodos.

Mnemónico	***Clase***			
CC-1	Documento			
Atributos				
Nombre	***Tipo***	***Valor por defecto***	***Semántica***	***Visibilidad***
contenido	String	null	Se refiere al contenido del documento de trabajo	Private
Estado	String	Inactivo	Estado en el que se encuentra el documento	Private
Nombre	String	null	Nombre por el cual se identifica el documento de trabajo.	Private
Tipo	String	rarq	Tipo de documento; generalmente identificado por la extensión que maneja.	Prívate
Ubicación	String	null	Ruta en el directorio del pc en donde se encuentra ubicado el documento	Prívate
Métodos				
Nombre	***Parámetros (tipo)***	***Tipo de retorno***	***Semántica***	
getContenido	-	String	Retorna el contenido de un objeto de tipo documento	
setContenido	cotenido:String	void	Establece el valor del contenido para un objeto de tipo documento	
getEstado	-	String	Retorna el estado de un objeto de tipo documento	

Mnemónico	*Clase*		
CC-1	Documento		
setEstado	estado:String	void	Establece el valor del estado para un objeto de tipo documento
getNombre	-	String	Retorna el nombre de un objeto de tipo documento
setNombre	nombrePath:String	void	Establece el valor del nombre para un objeto de tipo documento
getTipo	-	String	Retorna el tipo de un objeto de tipo documento
setTipo	tipo:String	void	Establece el valor del tipo para un objeto de tipo documento
getUbicacion	-	String	Retorna la ubicación de un objeto de tipo documento
setUbicacion	ubicacion:String	void	Establece la ubicación de un objeto de tipo documento

Tabla 7.2: Especificación Clase CC-1 Documento

Mnemónico	*Clase*			
CC-2	Validador			
Atributos				
Nombre	***Tipo***	***Valor por defecto***	***Semántica***	***Visibilidad***
documentoValido	boolean	false	Indicador que señala si el documento es válido o no.	Private
documento	Documento	null	Objeto que se recibe para validar su contenido	Private
factory	SchemaFactory	null	Objeto que lee representaciones externas de esquemas y los prepara para realizar la validación del contenido XML.	Private
schemaLocation	File	null	Representación abstracta de un archivo o directorio usando	Private

Mnemónico	*Clase*			
CC-2	Validador			
			sus rutas físicas. Se usa para ubicar el esquema físico.	
schema	Schema	null	Objeto que representa un esquema físico para la validación de un contenido XML.	Private
validator	Validator	null	Procesador que chequea un documento XML contra un esquema	Private
source	Source	null	Objeto que representa la fuente de datos, en este caso, un documento XML	Private

Métodos			
Nombre	***Parámetros (tipo)***	***Tipo de retorno***	***Semántica***
Validar	contenidoArchivo: String, file: File	Documento	Verifica si el documento XML es válido con respecto a los esquemas de validación correspondientes.
isDocumentoValido	-	Boolean	Retorna el valor del atributo documentoValido para un objeto de tipo Validador
setDocumentoValido	documentoValido: Boolean	void	Establece el valor del atributo documentoValido para un objeto de tipo Validador
getDocumento	-	Documento	Retorna el valor del atributo documento para un objeto de tipo Validador
setDocumento	documento: Documento	void	Establece el valor del atributo documento para un objeto de tipo Validador

Tabla 7.3: Especificación Clase CC-2 Validador

Mnemónico	Clase			
CC-3	Constantes			
Atributos				
Nombre	**Tipo**	**Valor por defecto**	**Semántica**	**Visibilidad**
color	Color	null	Color de visualización de los documentos en la aplicación dependiendo de su estado (activo, inactivo)	Public
extensionesBasicas	String	null	Extensiones que acepta el traductor (.xml y .rarq)	Public
rutaUtilidades	String	null	Ruta donde se encuentran ubicados todos los elementos necesarios para el funcionamiento del traductor	Public
Métodos				
Nombre	**Parámetros (tipo)**	**Tipo de retorno**	**Semántica**	
-	-	-	-	

Tabla 7.4: Especificación Clase CC-3 Constantes

Mnemónico	Clase			
CC-4	Interprete			
Atributos				
Nombre	**Tipo**	**Valor por defecto**	**Semántica**	**Visibilidad**
Formula	TeXFormula	null	Representa las sentencias en código LaTeX del documento traducido	Private
documento	Documento	null	Objeto que contiene la información a ser interpretada	Private
Métodos				
Nombre	**Parámetros (tipo)**	**Tipo de retorno**	**Semántica**	
interpretar	documento:Documento	JLabel	Se encarga de expresar el contenido en código LaTeX del documento traducido para transformarlo en una imagen	

Mnemónico	*Clase*		
CC-4	Interprete		
			que muestra la respectiva interpretación
getFormula	-	TeXFormula	Retorna el valor del atributo formula para un objeto de tipo Interprete
getDocumento	-	Documento	Retorna el valor del atributo documento para un objeto de tipo Interprete
setDocumento	documento:Documento	void	Establece el valor del atributo documento para un objeto de tipo Interprete
setFormula	formula:TeXFormula	void	Establece el valor del atributo formula para un objeto de tipo Interprete

Tabla 7.5: Especificación Clase CC-4 Intérprete

Mnemónico	*Clase*			
CC-5	Traductor			
Atributos				
Nombre	*Tipo*	*Valor por defecto*	*Semántica*	*Visibilidad*
documentoTraducido	Documento	null	Objeto que representa un documento que ha sido traducido	Private
tFactory	Transformer Factory	null	Objeto que permite la construcción del transformador XSL	Private
transformer	Transformer	null	Objeto que se usa para realizar la transformación de un documento XML	Private
Writer	StringWriter	null	Objeto que construye una cadena de caracteres de la transformación a	Private

Mnemónico	***Clase***			
CC-5	Traductor			
			partir de la entrada de un flujo de datos de un documento XML	
Reader	StringReader	null	Objeto que construye un flujo de datos a partir del contenido del documento XML que es una cadena de caracteres.	Private
Result	Result	null	Objeto que actúa como un contenedor para el resultado de la transformación de un documento XML	Private
Source	StreamSource	null	Objeto que actúa como contenedor de la fuente de datos para realizar la transformación de un documento XML	Private

Métodos			
Nombre	***Parámetros (tipo)***	***Tipo de retorno***	***Semántica***
Traducir	documentoImportado: Documento	Documento	Se encarga de tomar el contenido del documento importado y convertirlo en código LaTeX, de acuerdo con la sintaxis del cálculo $-p_{arq}$
getDocumentoTraducido	-	Documento	Retorna el valor del atributo

Mnemónico	*Clase*		
CC-5	Traductor		
			documentoTraducido para un objeto de tipo Traductor
setDocumentoTraducido	documentoTraducido: Documento	void	Establece el valor del atributo documentoTraducido para un objeto de tipo Traductor

Tabla 7.6: Especificación Clase CC-5 Traductor

Mnemónico	*Clase*			
CC-6	TeXFormula[2]			
Atributos				
Nombre	*Tipo*	*Valor por defecto*	*Semántica*	*Visibilidad*
-	-	-	-	-
Métodos				
Nombre	*Parámetros (tipo)*	*Tipo de retorno*	*Semántica*	
Add	s:String	TeXFormula	Analiza la cadena de caracteres dada e inserta la fórmula resultante al final de la actual TeXFormula	
createTeXIcon	size:float, style:int	TeXIcon	Crea un TeXIcon de esta TeXFormula usando el TeXFont por defecto en el tamaño dado y a partir del estilo TeX dado.	

Tabla 7.7: Especificación Clase CC-6 TeXFormula

Mnemónico	*Clase*			
CC-7	File[3]			
Atributos				
Nombre	*Tipo*	*Valor por defecto*	*Semántica*	*Visibilidad*

[2] Para la clase TeXFormula se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.8

[3] Para la clase File se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.1.

Mnemónico	*Clase*		
CC-7	File[3]		
Métodos			
Nombre	*Parámetros (tipo)*	*Tipo de retorno*	*Semántica*
isDirectory	-	boolean	Prueba que el archivo que invoca este método es un directorio
getName	-	String	Obtiene el nombre del objeto que invoca este método
getAbsoluteFile	-	File	Retorna la ruta completa del objeto que invoca éste método
getCanonicalPath	-	String	Retorna la cadena completa de la ruta del objeto que invoca éste método.

Tabla 7.8: Especificación Clase CC-7 File

Mnemónico	*Clase*			
CC-8	TransformerFactory[4]			
Atributos				
Nombre	*Tipo*	*Valor por defecto*	*Semántica*	*Visibilidad*
-	-	-	-	-
Métodos				
Nombre	*Parámetros (tipo)*	*Tipo de retorno*	*Semántica*	
newInstance	-	TransformerFactory	Obtiene una nueva instancia de un objeto de tipo TransformerFactory	
newTransformer	streamSource: StreamSource	Transformer	Crea un nuevo objeto transformador a partir de un flujo de datos que el cual es una representación del archivo XSLT.	

Tabla 7.9: Especificación Clase CC-8 TransformerFactory

[4] Para la clase TransformerFactory se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.9.

Mnemónico	Clase			
CC-9	SchemaFactory[5]			
Atributos				
Nombre	**Tipo**	**Valor por defecto**	**Semántica**	**Visibilidad**
-	-	-	-	-
Métodos				
Nombre	**Parámetros (tipo)**	**Tipo de retorno**	**Semántica**	
newInstance	schemaLanguage: String	SchemaFactory	Busca una implementación del SchemaFactory que soporte el lenguaje del esquema especificado y lo retorne.	
newSchema	schema:File	Schema	Analiza gramaticalmente el archivo especificado como un esquema y lo retorna como esquema.	

Tabla 7.10: Especificación Clase CC-9 SchemaFactory

Mnemónico	Clase			
CC-10	Source[6]			
Atributos				
Nombre	**Tipo**	**Valor por defecto**	**Semántica**	**Visibilidad**
-	-	-	-	-
Métodos				
Nombre	**Parámetros (tipo)**	**Tipo de retorno**	**Semántica**	
getSystemId	void	String	Obtiene el identificador del sistema que fue determinado con setSystemId.	
setSystemId	systemId:String	void	Determina el identificador del sistema para el Source.	

Tabla 7.11: Especificación Clase CC-10 Source

[5] Para la clase SchemaFactory se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.3.

[6] Para la interface Source relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.11.

Mnemónico	Clase			
CC-11	Validator[7]			
Atributos				
Nombre	**Tipo**	**Valor por defecto**	**Semántica**	**Visibilidad**
-	-	-	-	-
Métodos				
Nombre	**Parámetros (tipo)**	**Tipo de retorno**	**Semántica**	
Reset	void	void	Restablece este validador a su configuración original.	
Valídate	source:Source	void	Valida la entrada especificada.	
Valídate	result:Result, source:Source	void	Valida la entrada especificada y envía la validación ampliada a la salida determinada.	

Tabla 7.12: Especificación Clase CC-11 Validator

Mnemónico	Clase		
CC-12	StringWriter[8]		
Atributos			
Nombre	**Tipo**	**Valor por defecto**	**Semántica**	**Visibilidad**
-	-	-	-	-
Métodos			
Nombre	**Parámetros (tipo)**	**Tipo de retorno**	**Semántica**
Close	void	void	Cerrar el StringWritter no tiene ningún efecto.
Flush	void	void	Limpia la secuencia
getBuffer	void	StringBuffer	Retorna la misma cadena de caracteres buffer.
toString	void	String	Retorna el valor del actual buffer como una cadena de caracteres.
write	len:int, off:int, cbuf:char[]	void	Escribe una porción de un arreglo de caracteres
write	str:String	void	Escribe una cadena de caracteres
write	len:int, off:int, str:String	void	Escribe una porción de una cadena de caracteres

Tabla 7.13: Especificación Clase CC-12 StringWriter

[7] Para la clase Validator se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.10.

[8] Para la clase StringWriter se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.7.

Mnemónico	Clase			
CC-13	StringReader[9]			
Atributos				
Nombre	**Tipo**	**Valor por defecto**	**Semántica**	**Visibilidad**
-	-	-	-	-
Métodos				
Nombre	**Parámetros (tipo)**	**Tipo de retorno**	**Semántica**	
close	void	void	Cierra el flujo	
mark	readAheadLimit: int	void	Marca la presente posición en el flujo.	
markSupported	void	void	Dice si este flujo soporta la operación mark(), la cual lo hace.	
read	void	int	Lee un único carácter.	
read	cbuf:char[], off:int, len:int	int	Lee caracteres en una porción de un arreglo.	
ready	void	boolean	Dice si el flujo está listo para ser leído.	
reset	void	void	Restablece el flujo a la marca más reciente, o al comienzo de la cadena de caracteres si este no ha sido marcado.	
skip	ns:long	long	Salta el número de caracteres especificado en el flujo.	

Tabla 7.14: Especificación Clase CC-13 StringReader

Mnemónico	Clase			
CC-14	Transformer[10]			
Atributos				
Nombre	**Tipo**	**Valor por defecto**	**Semántica**	**Visibilidad**
-	-	-	-	-
Métodos				
Nombre	**Parámetros (tipo)**	**Tipo de retorno**	**Semántica**	
clearParameters	Void	void	Limpia todos los parámetros determinados con setParameter	

[9] Para la clase StringReader se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.6.

[10] Para la clase Transformer se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.12.

Mnemónico	*Clase*		
CC-14	Transformer[10]		
transform	outputTarget:Result, xmlSource:Source	void	Transforma el código fuente XML a un resultado.

Tabla 7.15: Especificación Clase CC-14 Transformer

Mnemónico	*Clase*			
CC-15	StreamSource[11]			
Atributos				
Nombre	*Tipo*	*Valor por defecto*	*Semántica*	*Visibilidad*
-	-	-	-	-
Métodos				
Nombre	*Parámetros (tipo)*	*Tipo de retorno*	*Semántica*	
getInputStream	void	InputStream	Obtiene el byte del flujo que fue establecido con setByteStream	
getPublicId	void	String	Obtiene el identificador publico que fue establecido con setPublicId	
getReader	void	Reader	Obtiene el carácter del flujo que fue establecido con setReader	
getSystemId	void	String	Obtiene el identificador del sistema que fue establecido con setSystemId	
setInputStream	inputStream: InputStream	Void	Establece el byte del flujo para ser usado como una entrada	
setPublicId	publicId:String	Void	Establece el identificador público para este Source.	
setReader	reader:Reader	Void	Establece la entrada para ser un lector de carácter.	
setSystemId	file:File	Void	Establece el ID del sistema desde un archivo de referencia.	
setSystemId	systemId:String	Void	Establece el identificador del sistema para este Source.	

Tabla 7.16: Especificación Clase CC-15 StreamSource

[11] Para la clase StreamSource se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.5.

Mnemónico	*Clase*			
CC-16	Result[12]			
Atributos				
Nombre	*Tipo*	*Valor por defecto*	*Semántica*	*Visibilidad*
-	-	-	-	-
Métodos				
Nombre	*Parámetros (tipo)*	*Tipo de retorno*	*Semántica*	
getSystemId	void	String	Obtiene el identificador del sistema que fue establecido con setSystemId	
setSystemId	systemId:String	void	Establece el identificador del sistema para este resultado.	

Tabla 7.17: Especificación Clase CC-16 Result

Mnemónico	*Clase*			
CC-17	Schema[13]			
Atributos				
Nombre	*Tipo*	*Valor por defecto*	*Semántica*	*Visibilidad*
-	-	-	-	-
Métodos				
Nombre	*Parámetros (tipo)*	*Tipo de retorno*	*Semántica*	
newValidator	void	Validador	Crea un nuevo validador para este esquema.	
newValidatorHandler	void	ValidatorHandler	Crea un nuevo manejador de validación para este esquema.	

Tabla 7.18: Especificación Clase CC-17 Schema

[12] Para la clase Result se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.13.

[13] Para la clase Schema se relacionan en el diccionario de clases todos los métodos y atributos que son usados en la aplicación, los demás podrán ser consultados en el anexo 15.2.2.

7.3 RECURSOS PERSISTENTES

Además de las clases propuestas en el modelo estructural, el sistema usa algunos documentos que en tiempo de ejecución son persistentes. A continuación se detalla cada uno de éstos:

7.3.1 Esquema xml transformerumlarchro (xml schema)

El sistema hace uso de los esquemas desarrollados por la Universidad de California, Irvine en el proyecto xADL 2.0 extendido [ISR, 2008]. Específicamente los esquemas instance.xsd, types.xsd, boolguard.xsd y options.xsd. Sin embargo, además de usar dichos esquemas es necesario extenderlos para que logren soportar conceptos claves del cálculo formal – ρ_{arq}. El esquema que se usará para lograr una extensión básica es el umlarchro.xsd [Arquisoft, 2008]. Adicionalmente, con el objetivo de soportar algunos conceptos específicos de la sintaxis del cálculo formal – ρ_{arq}, se desarrolló una extensión al umlarchro.xsd el cual se denomina transformerUmlarchro.xsd.

A continuación se describe la estructura detallada del esquema extendido transformerUmlarchro.xsd, la cual debe ser respetada por los documentos XML de entrada:

- Elemento Raíz *transforUmlstructure:transforArchRo*: Elemento principal para la definición de la Arquitectura. Contiene la descripción de la arquitectura propuesta, esta compuesto por los elementos *TransforComponentRo*, *OptionalConnectorRo* y *connectorRo* que pueden tener cero o muchas ocurrencias, y una

descripción de la Arquitectura (El elemento *connectorRo* pertenece al esquema umlarcrho.xsd [Arquisoft, 2008]).

- Elemento *TransforComponentRo*: Extiende al "*complexType*" "*ComponentRo*" del esquema "umlarchro.xsd", heredando todos los elementos contenidos en él. Se añade el elemento RolRo para manejar los roles de un componente dentro de una arquitectura (initial, errorHandler y final). Tiene también elementos internos del tipo *TransformBoolEval* y *VariableSet.*
- Elemento *RolRo*: Contiene el posible tipo de rol (initial, errorHandler, final) que se especifica al definir el elemento transforComponentRo. Toma los valores definidos en el simpleType *RolRoSimple*.
- Elemento *RolRoSimple:* Especifica los roles de un componente en una arquitectura, definidos según la especificación del cálculo arquitectural Ro (arq).
- Elemento *TransformBoolEva:* Elemento para controlar el flujo de información de la arquitectura validando el éxito o fracaso de la ejecución de cada componente. Contiene dos elementos de tipo *ReferenceRo*.
- Elemento *VariableSet:* Determina un conjunto de variables de un componente en una arquitectura, definidos según la especificación del cálculo formal – ρ_{arq} para identificar componentes condicionados o alternativos.
- Elemento *ReferenceRo:* Elemento que referencia las entidades a las cuales se redirecciona el flujo de informacion después de la validación del éxito o fracaso de la ejecución de un componente anterior. Tiene un elemento interno de tipo Estado.
- Elemento Estado: Elemento que describe el estado de un componente en ejecución. Varia entre *true* y *false.*

- Elemento *OptionalConnectorRo:* Elemento que representa un conector entre dos componente que solo se activara de acuerdo a una restricción arquitectural dada. Contiene un elemento interno de tipo *BooleanGuard* que pertenece al esquema boolguard.xsd [ISR, 2008].

La estructura y el código fuente del esquema tranformerUmlarchro.xsd pueden ser consultados en documentación/anexos/ANEXO5 y documentación/anexos/ANEXO3 respectivamente en el CD adjunto a este informe.

7.3.2 Hoja de estilos transformerumlarchroxsl

Existen dos diferentes lenguajes de estilos recomendados por la W3C para dar formato a un documento XML: las hojas de estilo en cascada (CSS) y el lenguaje extendible de hojas de estilo (XSL). Según la misma W3C, las hojas de estilo en cascada son mucho más sencillas de utilizar, sin embargo, admiten que el lenguaje extendible de hojas de estilo es mucho más potente, así que resumen su recomendación en la siguiente frase: "*use CSS when you can, use XSL when you must*" [W3C, 1, 2008] (Usa CSS cuando puedas, usa XSL cuando debas).

El sistema hace uso de una hoja XSL que permite realizar la transformación de un documento XML de entrada. Esta hoja se denomina transformerumlarchroxsl.xsl. La estructura y el código fuente de la hoja transformerumlarchroxsl.xsl pueden ser consultados en documentación/anexos/ANEXO4 y documentación/anexos/ANEXO3 respectivamente en el CD adjunto a este informe.

8 MODELO DINÁMICO

En esta parte del documento se presentan los aspectos de comportamiento del sistema que cambian con el tiempo a través de diagramas de secuencia que describen la interacción en el tiempo de los objetos dentro del sistema, por cada caso de uso identificado en el modelo funcional.

Con éste modelo se pretende especificar e implementar los aspectos de control del sistema permitiendo describir las secuencias de las operaciones que se producen a través del tiempo.

8.1 DIAGRAMA DE SECUENCIA CU-3 CERRAR APLICACIÓN

8.2 DIAGRAMA DE SECUENCIA CU-3.1 CERRAR ARCHIVO

8.3 DIAGRAMA DE SECUENCIA CU-4 IMPORTAR ARCHIVO

8.4 DIAGRAMA DE SECUENCIA CU-4.1 VALIDAR ARCHIVO

8.5 DIAGRAMA DE SECUENCIA CU-5 TRADUCIR ARCHIVO

8.6 DIAGRAMA DE SECUENCIA CU-5.1 MOSTRAR ARCHIVO TRADUCIDO

8.7 DIAGRAMA DE SECUENCIA CU-6 GUARDAR ARCHIVO TRADUCIDO

9 DISEÑO BÁSICO DE LA INTERFAZ GRÁFICA DEL USUARIO

9.1 DIAGRAMA DE NAVEGACIÓN

Este diagrama permite dar una vista general acerca de cómo será la navegabilidad a través del aplicativo:

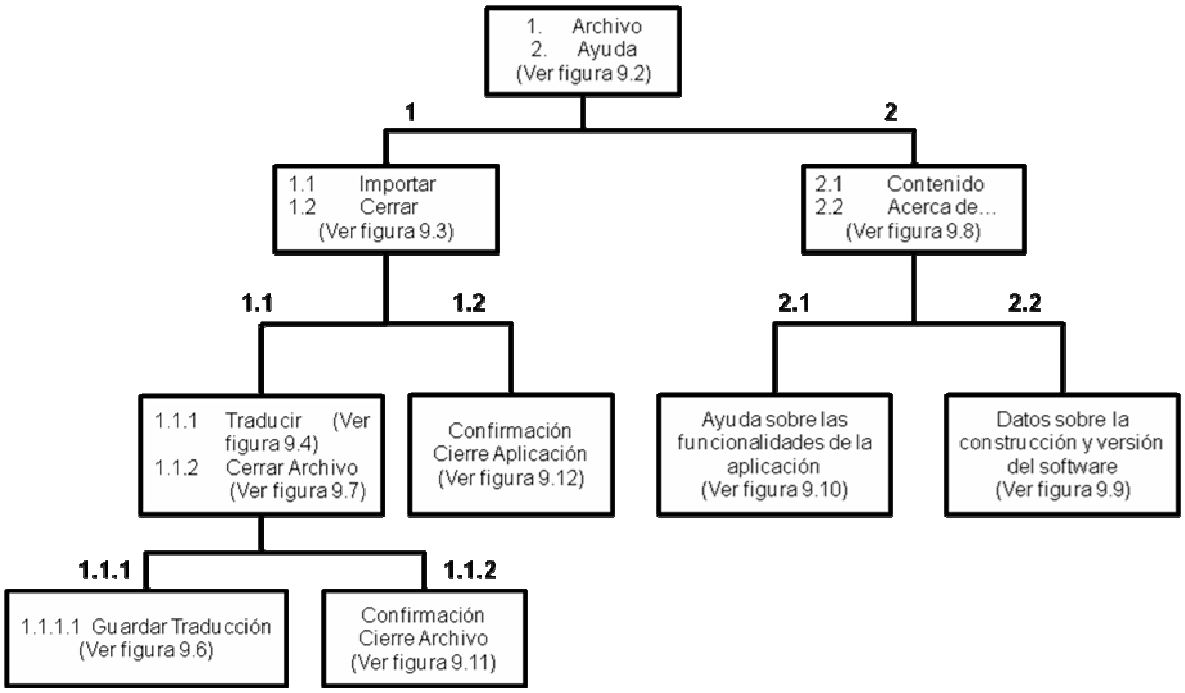

Figura 9.1 Diagrama de Navegación

9.2 BOCETOS DE EXPOSICIONES DE PANTALLA

A continuación se muestran las exposiciones por pantalla correspondientes al prototipo básico de la aplicación:

- Pantalla principal:

Figura 9.2: Pantalla Principal de la Aplicación

- Menú Archivo:

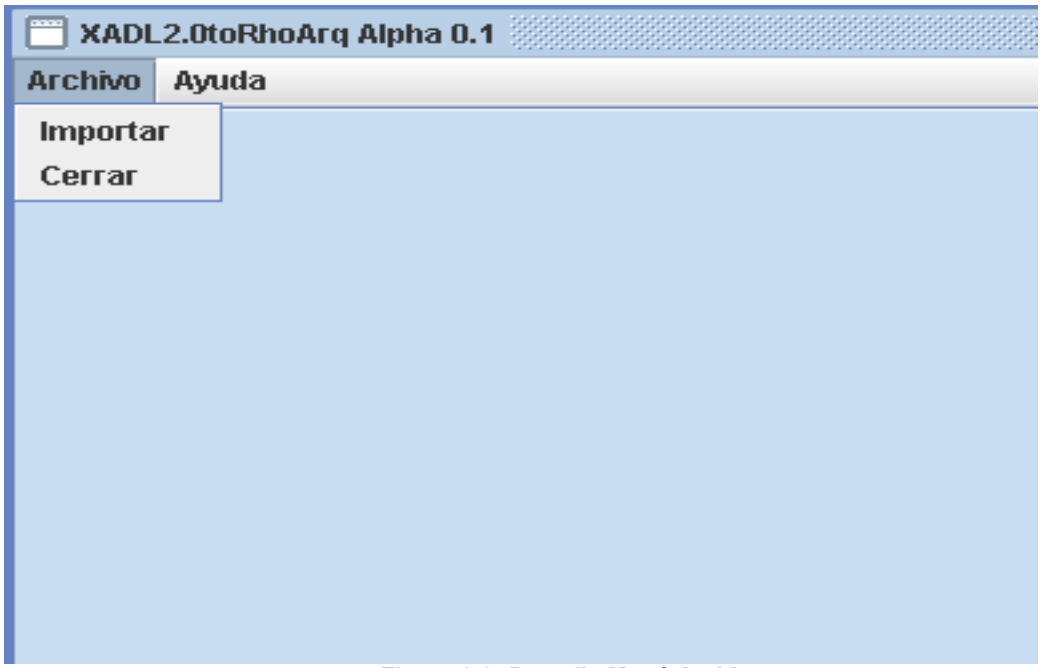

Figura 9.3: Pantalla Menú Archivo

- Marco de trabajo con un archivo importado:

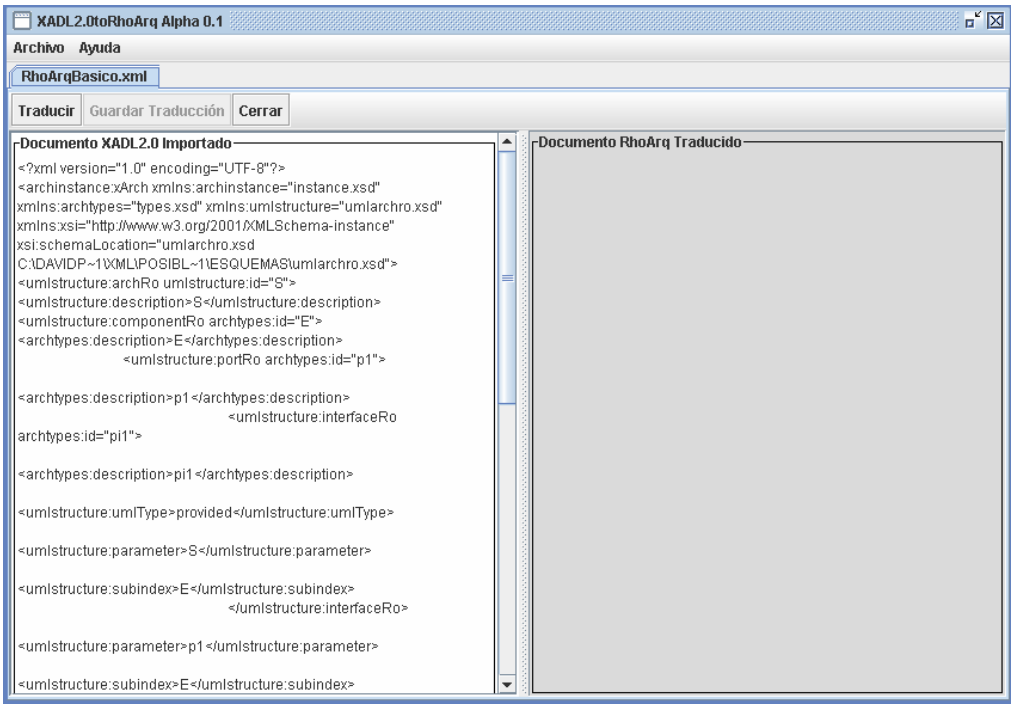

Figura 9.4: Pantalla Archivo Importado

- Marco de trabajo con varios archivos importados:

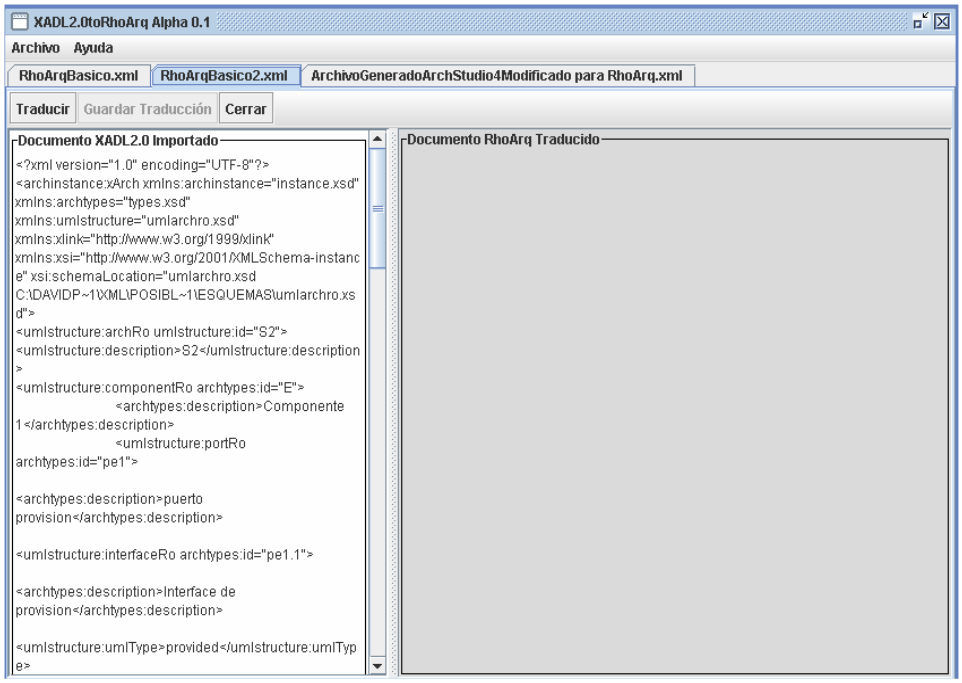

Figura 9.5: Pantalla Importación Archivos

- Marco de trabajo con un archivo traducido:

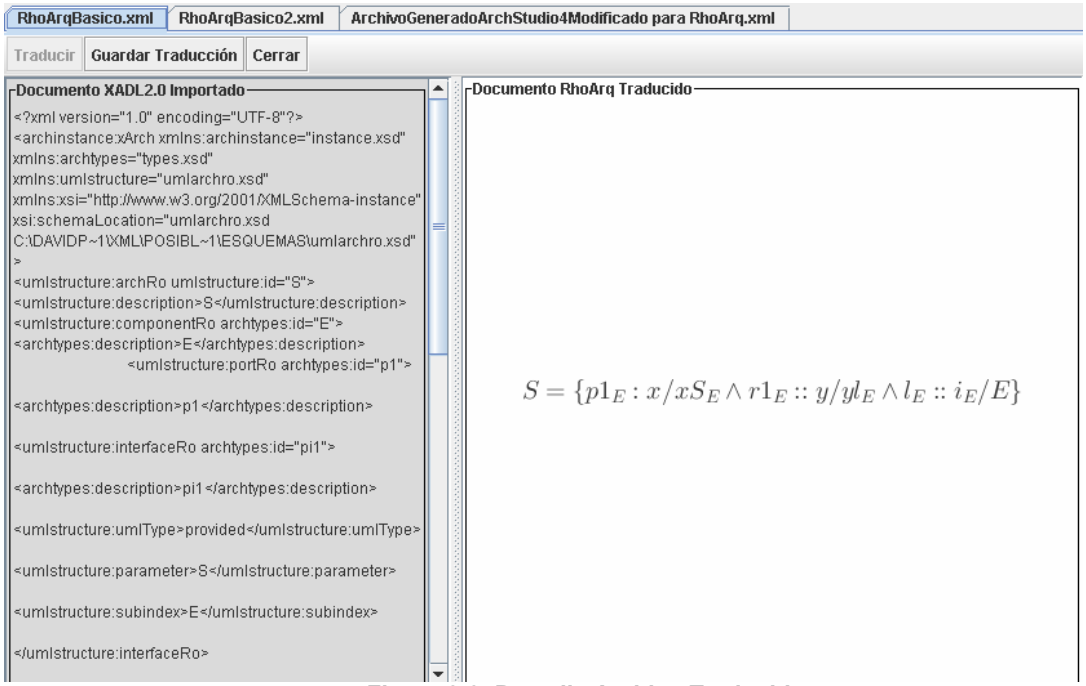

Figura 9.6: Pantalla Archivo Traducido

- Marco de Trabajo con un archivo que fue traducido y guardado:

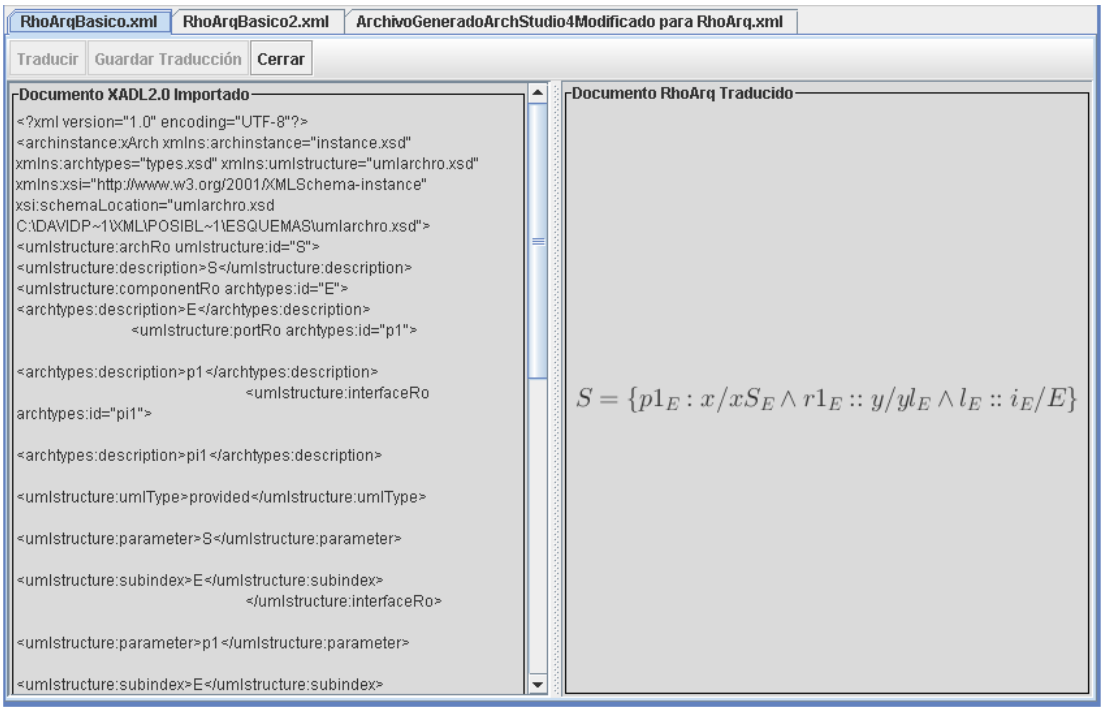

Figura 9.7: Pantalla Archivo Traducido Guardado

- Menú Ayuda:

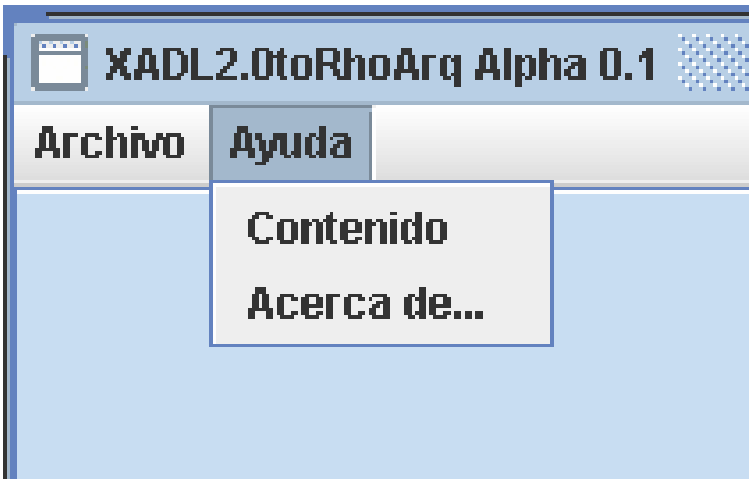

Figura 9.8: Pantalla Menú Ayuda

- Ventana Acerca de:

Figura 9.9: Pantalla Menú Acerca de

- Ventana Contenido Ayuda:

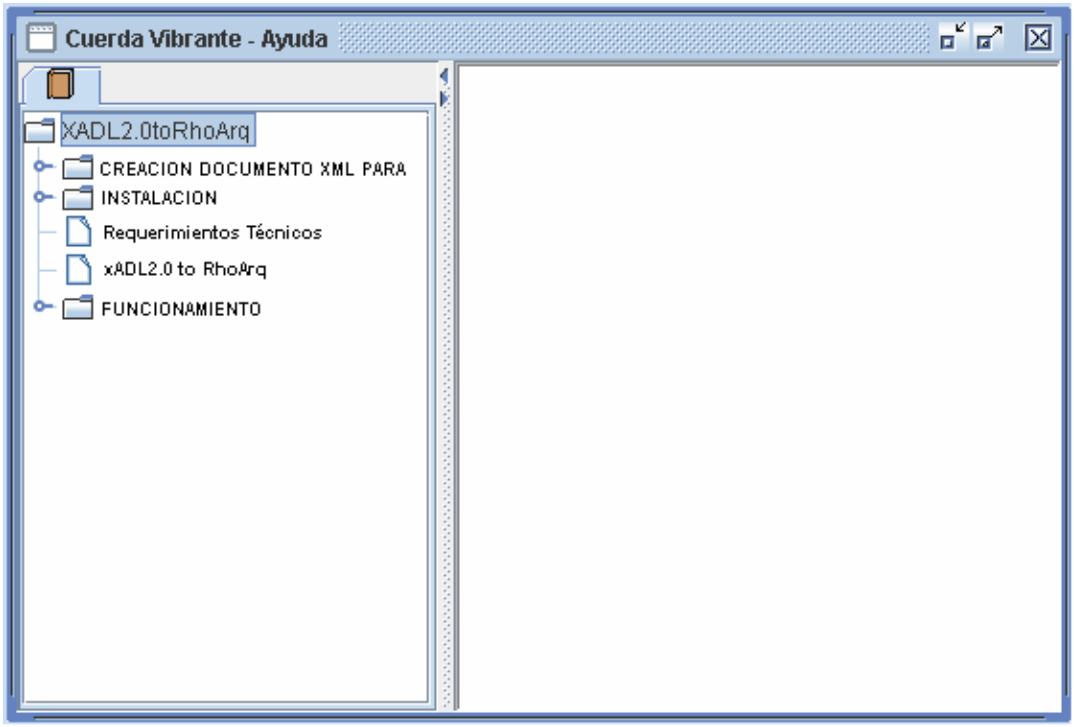

Figura 9.10: Pantalla Contenido Ayuda

- Ventana Cerrar Archivo:

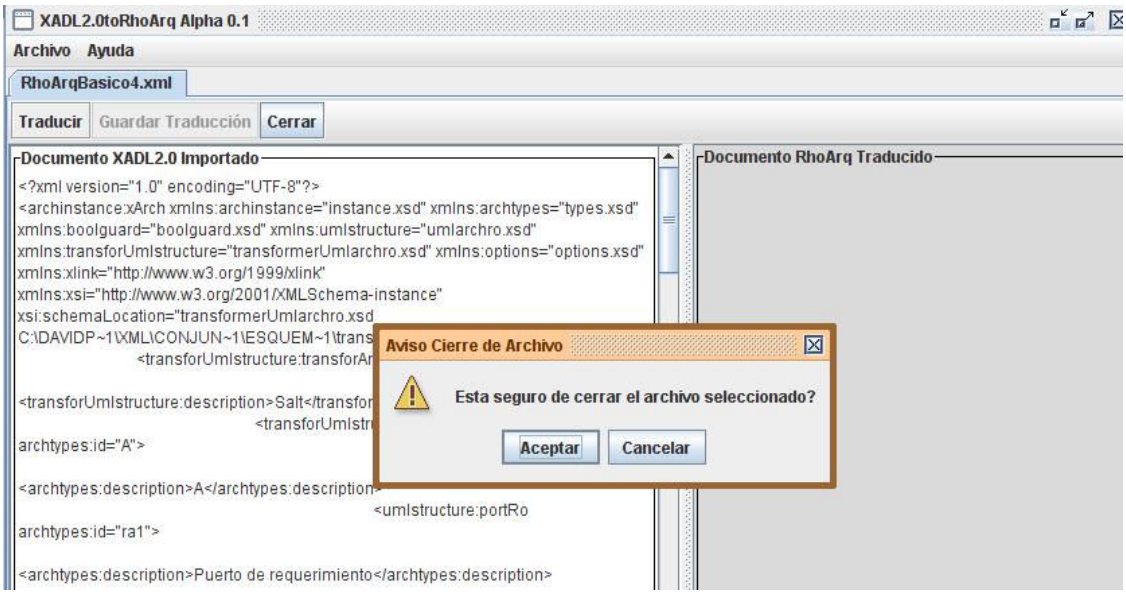

Figura 9.11: Pantalla Cerrar Archivo

- Ventana Cerrar Aplicación:

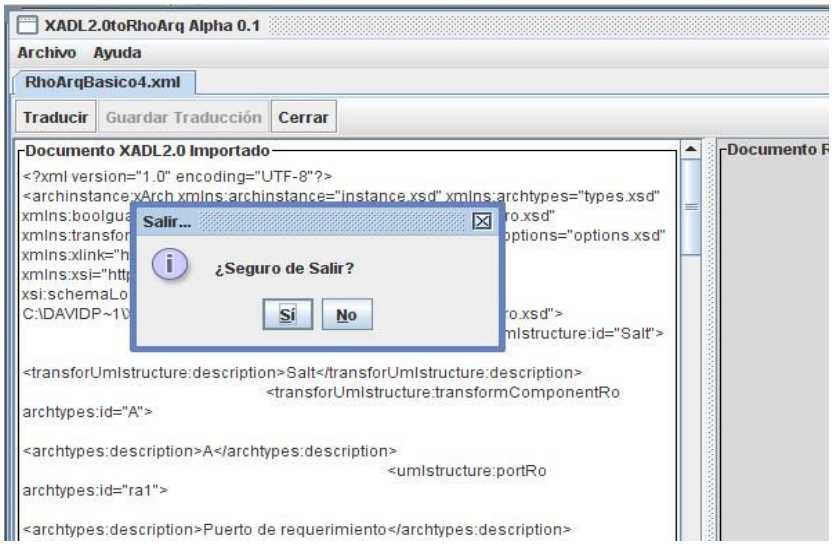

Figura 9.12: Pantalla Cerrar Aplicación

10 PROYECTOS RELACIONADOS

Para la realización del proyecto aquí presentado, fue necesaria la comprensión y el uso de otros proyectos externos. A continuación se muestran cada uno de ellos, exponiendo su estructura básica y la forma en la cual se adaptaron y unieron dentro del proyecto expuesto en este documento:

10.1 ESQUEMA UMLARCHRO

Para el proceso básico de validación de documentos XML, era necesaria la construcción de un esquema que soportara los conceptos del cálculo para el modelamiento formal de arquitecturas de software – ρ_{arq}. Sin embargo, el proyecto “Traducción de especificaciones arquitecturales de software en xADL2.0 estereotipadas a formato de visualización factorial SVG” presentado

por Ana Milena Parra y Juan Carlos Reyes [Arquisoft, 2008], tenía ya un esquema base, sobre el cual se podían soportar algunos conceptos del cálculo de especificación formal para el modelamiento de arquitecturas de software – *Para*. A partir de este se esquema se construye el esquema final transformerUmlarchro.xsd, el cual es una extensión del umlarchro.xsd, que fue el desarrollado en el proyecto realizado por Ana Milena Parra y Juan Carlos Reyes.

10.2 xADL 2.0

Como se mencionó en la sección 4.8 para éste proyecto se hizo uso de xADL 2.0, el cual fue desarrollado por la Universidad de Irvine. Este Lenguaje de Descripción Arquitectural (LDA) es altamente extendible ya que está basado en XML [ISR, 2008]. De este proyecto se utilizaron en particular los esquemas: instance.xsd, types.xsd, boolguard.xsd y options.xsd.

Específicamente fue necesario utilizar éstos esquemas para el proceso de validación de los documentos XML de entrada. De estos 4 esquemas, el instance.xsd, el options.xsd y el types.xsd se usan en el esquema umlarchro.xsd extendiendo algunos de los elementos propios de cada uno de ellos. Del esquema boolguard.xsd se extendieron algunos elementos necesarios para definir componentes opcionales dentro de una arquitectura.

10.3 PROYECTO JMATHTEX

Con el fin de mostrar el resultado final de la transformación de un documento de entrada XML con la especificación XADL2.0, a un documento con sintaxis de LaTeX, se hizo necesario el uso de parte del proyecto JMathTex [Cleemput et al., 2008]. JMathTeX es una librería que provee un conjunto de clases, que

permite mostrar formulas matemáticas complejas, como parte de una aplicación Java. En particular, en el desarrollo de la aplicación, se uso la clase TeXFormula, para la cual podemos ver la documentación completa en el anexo 15.2.8.

11 ASPECTOS DE IMPLEMENTACIÓN

En esta sección se mostrará la tecnología usada en el desarrollo de la aplicación, y las diferentes capas que fueron creadas para lograr el cumplimento de los requerimientos expuestos al principio de este documento (Ver Sección 5.3). Además se menciona el software que fue necesario para realizar el proyecto, sus pruebas y diferentes recursos que son indispensables para su funcionamiento.

11.1 TIPO DE TECNOLOGÍA

Las tecnologías usadas en el desarrollo de la aplicación involucran el lenguaje de programación (JAVA), XML y XSLT. Además también se uso un conjunto de clases especializadas en el trabajo entre JAVA, XML y Latex para realizar los procesos de validación y transformación de documentos de entrada:

- JAVA: La aplicación fue desarrollada sobre la plataforma J2SE (Java 2 Standard Edition) en su versión 6, usando también la versión 6 del JRE (Java Run Enviroment) [SUN, 2008]. La interfaz de desarrollo usada fue Eclipse en su versión 3.3 [Eclipse, 2008].

- XML: XML es un lenguaje derivado del SGML (Standard Generalized Markup Language) y busca describir documentos y datos de una forma estandarizada basándose en un formato de texto, el cual puede ser

fácilmente interpretado y transportado a través de protocolos de internet [BBJD, 2003]. En este proyecto, al usar un LDA como XADL2.0, fue necesario aprender y comprender los conceptos de XML. Esto implica saber si un documento XML está bien formado, como construir un esquema XML y como validar un documento contra ese esquema construido. La versión de XML utilizada fue la 1.0 [W3C, 2008].

- XSL: Ya en la sección 4.7 se explico la estructura básica y funcionamiento de las XSL, y en la sección 7.3.2 se explico la XSL resultante en este proyecto. Las versiones de la especificación para realizar la XSL resultante es la versión 1.0 [W3C, 2008].

- Validación y Transformación XML en Java: Para realizar los procesos de validación y transformación XML en Java, fue necesario el uso de clases especializadas agrupadas de la siguiente forma:

    - Javax.xml.validation:
        - Schema (Ver anexo 15.2.2)
        - SchemaFactory (Ver anexo 15.2.3)
        - Validator (Ver anexo 15.2.10)

    - Javax.xml.transform:
        - Source (Ver anexo 15.2.11)
        - StreamSource (Ver anexo 15.2.5)
        - Result (Ver anexo 15.2.13)
        - StreamResult (Ver anexo 15.2.4)
        - Transformer (Ver anexo 15.2.12)
        - TransformerFactory (Ver anexo 15.2.9)

- Librería externa para el trabajo con formulas matemáticas en Latex: Para realizar la generación y el despliegue del resultado de las transformaciones de los documentos XML, fue necesario usar clases externas que mostraran símbolos matemáticos y que además generaran documentos de extensión .tex. En la sección 10.3 podemos encontrar la información correspondiente al proyecto JMathTex de donde provienen las clases usadas en el proyecto.

11.2 SOFTWARE EXTERNO UTILIZADO

Se utilizaron diferentes herramientas de software para la realización del proyecto. A continuación se enumeran cuales fueron y el porque de su utilización:

- Eclipse Versión 3.3: Es el entorno de desarrollo utilizado para codificar la aplicación. Puede ser descargado de manera gratuita en http://www.eclipse.org/downloads/.

- Altova XMLSpy Proffessional Edition Versión 2008: Es un editor XML que provee un ambiente de desarrollo para modelar, codificar, transformar y depurar tecnologías relacionadas con XML. La herramienta puede ser descargada en http://www.altova.com/download.html pero tiene una duración limitada. Se utilizó par la creación del esquema transformerUmlarchro.xsd y la hoja de estilos transformerUmlarchroxsl.xsl, además de servir para la construcción de los archivos XML para la realización de las pruebas funcionales.

- CreateInstallFree: Es una herramienta que facilita la creación de instaladores para sistemas operativos Windows. Puede ser descargada gratuitamente en http://createinstall-free.uptodown.com/. Se uso para generar el instalador de la aplicación de extensión .exe.

- BitRock InstallBuilder: Es una herramienta que facilita la creación de instaladores para sistemas operativos Windows, MacOS ó Linux. Puede ser descargada gratuitamente en http://bitrock.com/download_installbuilder_download_step2.html pero tiene una duración limitada. Se uso para generar el instalador de la aplicación para plataformas Linux que tiene extensión .bin.

- Enterprise Architec Versión 7.0: Es una herramienta que provee un ambiente para modelar sistemas de software usando UML 2.0 ó 2.1. Puede ser descargado gratuitamente en http://www.sparxsystems.com.au/products/ea/trial.html pero tiene una duración limitada. Se utilizó en la realización de todos los diagramas y modelos de la aplicación.

- ArchStudio 4: Es un ambiente de desarrollo de arquitecturas de software y se instala como un agregado de Eclipse 3.3. Puede ser gratuitamente descargado en http://www.isr.uci.edu/projects/archstudio/setup-easy.html. Se utilizo para la generación de archivos xADL 2.0 de diferentes tipos de arquitectura.

11.3 CONSTRUCCIÓN DEL PROTOTIPO FUNCIONAL

Como se mencionó en la sección 6.1, para éste proyecto se definió un estilo arquitectural estratificado en tres capas. A continuación se describirá la estructura para cada una de las capas que conforman esta arquitectura.

11.3.1 CAPA DE PRESENTACIÓN

- Clase BarraHerramientas: Esta clase fue desarrollada con el fin de crear una barra de herramientas por cada archivo de entrada a la aplicación. Cada barra tendría tres botones que dispararán los eventos de traducción, guardado y cerrado individual de cada archivo de entrada. El código fuente puede ser consultado en documentación/anexos/ANEXO1 el CD adjunto a este informe. La estructura de la clase se presenta a continuación:

Mnemónico	***Clase***			
CP-1	BarraHerramientas			
Atributos				
Nombre	***Tipo***	***Valor por defecto***	***Semántica***	***Visibilidad***
botonTraducir	JButton	null	Objeto grafico para disparar el evento de traducción.	private
botonExportarArchivo Traducido	JButton	null	Objeto grafico para disparar el evento de guardado.	private
botonCerrarArchivo	JButton	null	Objeto grafico para disparar el evento de cerrado del archivo.	private

Mnemónico	***Clase***			
CP-1	BarraHerramientas			
panelLocal	PanelPrincipal	null	Contenedor del archivo importado.	private
traductor	Traductor	null	Objeto de negocio que usa para realizar la traducción.	private
Métodos				
Nombre	***Parámetros (tipo)***		***Tipo de retorno***	***Semántica***
getBotonTraducir	-		JButton	Obtiene el objeto botón que dispara el evento de traducción.
setBotonTraducir	botonTraducir :JButton		-	Establece el objeto botón que dispara el evento de traducción.
getBotonExportarArchivoTraducido	-		JButton	Obtiene el objeto botón que dispara el evento de guardado.
setBotonExportarArchivoTraducido	botonExportarArchivoTraducido:JButton		-	Establece el objeto botón que dispara el evento de guardado.
getBotonCerrarArchivo	-		JButton	Obtiene el objeto botón que dispara el evento de cerrado individual de archivo.
setBotonCerrarArchivo	botonCerrarArchivo:JButton		-	Establece el objeto botón que dispara el evento de cerrado individual de archivo.
getPanelLocal	-		PanelPrincipal	Obtiene el objeto panelLocal

Mnemónico	*Clase*		
CP-1	BarraHerramientas		
setPanelLocal	panelLocal : PanelPrincipal	-	Establece el valor para el panelLocal
cambiarEstadoBotones	-	-	Habilita ó inhabilita ciertos botones dependiendo en que estado se encuentre el documento actual en el panelLocal.
inicializar	panel : PanelPrincipal	-	Se encarga de inicializar un objeto de tipo BarraHerramientas.

Tabla 11.1 Especificación Clase CP-1 BarraHerramientas

- Clase DialogoAcercaDe: Esta clase fue creada con el fin de crear un contenedor para los datos básicos de los creadores de la aplicación. La estructura de la clase se presenta a continuación:

Mnemónico	*Clase*			
CP-2	DialogoAcercaDe			
Atributos				
Nombre	*Tipo*	*Valor por defecto*	*Semántica*	*Visibilidad*
panelCentral	JPanel	null	Contenedor central del cuadro de dialogo.	private
labelNombre	JLabel	null	Muestra el nombre del aplicativo	private
labelLogo	JLabel	null	Muestra el logo del aplicativo	private
labelEscudo	JLabel	null	Muestra el escudo de la	private

Mnemónico	***Clase***			
CP-2	DialogoAcercaDe			
			Universidad Distrital Francisco José de Caldas.	
labelCreditos	JLabel	null	Muestra los datos de los creadores del aplicativo.	private
botonAceptar	JButton	null	Botón que dispara la acción de cerrar el cuadro de dialogo.	private
Métodos				
Nombre	***Parámetros (tipo)***		***Tipo de retorno***	***Semántica***
getPanelCentral	-		JPanel	Obtiene el valor del objeto panelCentral.
getBotonAceptar	-		JButton	Obtiene el valor del objeto botón que dispara la acción de cerrar el cuadro de dialogo.
inicializar	-		-	Inicializa todos los atributos de un objeto DialogoAcercaDe.

Tabla 11.2 Especificación Clase CP-2 DialogoAcercaDe

- Clase ExampleFileFilter: Esta clase fue usada para crear filtros de extensiones de archivo a la hora de importar y exportar archivos. El autor original de la clase es Jeff Dinkins y su completa especificacion puede ser vista en el anexo 15.2.16.

- Clase Imagen: Esta clase fue creada con el fin de facilitar el manejo de imágenes dentro de la aplicación. La estructura de la clase se presenta a continuación:

Mnemónico	***Clase***			
CP-3	Imagen			
Atributos				
Nombre	***Tipo***	***Valor por defecto***	***Semántica***	***Visibilidad***
-	-	-	-	-
Métodos				
Nombre	***Parámetros (tipo)***		***Tipo de retorno***	***Semántica***
crearImagen	name : String, cmp : java.awt.Component		Image	Recupera una imagen de los recursos de la aplicación y se la asigna a un componente como un JLabel ó un JButton.

Tabla 11.3 Especificación Clase CP-3 Imagen

- Clase PanelPrincipal: Es el contenedor de un documento que sea importado y que muestra la barra de herramientas para trabajar con el archivo. La estructura de la clase se presenta a continuación:

Mnemónico	***Clase***			
CP-4	PanelPrincipal			
Atributos				
Nombre	***Tipo***	***Valor por defecto***	***Semántica***	***Visibilidad***
splitPanel	JSplitPane	null	Es el contenedor del documento importado y del resultado de la traducción.	private

Mnemónico	***Clase***			
CP-4	PanelPrincipal			
areaDocumentoXADL	JTextArea	null	Es el área que contiene el contenido en texto plano del documento de entrada.	private
areaDocumentoRhoArq	JTextArea	null	Es el área que contiene el código latex de salida. Es invisible al usuario	private
miBarra	BarraHerramientas	null	Barra de herramientas que contiene los botones para traducir, guardar y cerrar un archivo importado.	private
scrollPanelBasico	JScrollPane	null	Contenedor del área del documento de entrada.	private
scrollPanelBasico2	JScrollPane	null	Contenedor del área del documento de salida.	private
documentoImportado	Documento	null	Objeto de negocio que representa a un documento de entrada.	private
documentoTraducido	Documento	null	Objeto de negocio que representa a un documento de salida.	private
ventanaPrincipal	VentanaPrincipal	null	Es la ventana principal del aplicativo.	private

Mnemónico	***Clase***			
CP-4	PanelPrincipal			
fileChooser	JFileChooser	null	Objeto que se usa para importar o guardar un archivo.	private
interprete	Interprete	null	Objeto de negocio que se encarga de traducir el código latex de salida y presentar una imagen con los símbolos matemáticos.	private
labelExpresion	JLabel	null	Objeto que contiene una imagen con símbolos matemáticos de salida.	private
index	int	0	Es el índice de la pestaña actual.	private

Métodos			
Nombre	***Parámetros (tipo)***	***Tipo de retorno***	***Semántica***
inicializar	indice : int, documento : Documento, documentoTraducido : Documento, ventanaLocal : VentanaPrincipal	-	Inicializa todos los atributos de un objeto de tipo PanelPrincipal.
getSplitPanel	-	JSplitPanel	Obtiene el objeto que contiene el documento importado y del resultado de la traducción
getAreaDocumentoXADL	-	JTextArea	Obtiene el área de texto que

Mnemónico	*Clase*		
CP-4	PanelPrincipal		
			contiene el documento de entrada.
getAreaDocumentoRhoArq	-	JTextArea	Obtiene el área de texto que contiene el documento de salida.
getMiBarra	-	BarraHerramientas	Obtiene la barra de herramientas actual.
getScrollPanelBasico	-	JScrollPane	Obtiene el objeto contenedor del área que tiene el documento de entrada.
setScrollPanelBasico	scrollPanelBasico : JScrollPane	-	Establece el valor para el contenedor del área que a su vez contiene el archivo de entrada.
getScrollPanelBasico2	-	JScrollPane	Obtiene el objeto contenedor del área que tiene el documento de salida.
setScrollPanelBasico2	scrollPanelBasico2 : JScrollPane	-	Establece el valor para el contenedor del área que a su vez contiene el archivo de salida.
getIndex	-	int	Obtiene el valor del índice de la pestaña actual.
setIndex	indice : int	-	Establece el valor del índice de la pestaña actual.

Mnemónico	***Clase***		
CP-4	PanelPrincipal		
getVentanaPrincipal	-	VentanaPrincipal	Obtiene el valor de la ventana del aplicativo actual.
setVentanaPrincipal	ventanaPrincipal : VentanaPrincipal	-	Establece el valor de la ventana del aplicativo actual
getDocumentoImportado	-	Documento	Obtiene el documento importado
setDocumentoImportado	documentoImportado : Documento	-	Establece el documento importado
getDocumentoTraducido	-	Documento	Obtiene el documento traducido.
setDocumentoTraducido	documentoTraducido : Documento	-	Establece el documento traducido.
getFileChooser	-	JFileChooser	Obtiene el valor del objeto grafico que importa un archivo.
setFileChooser	fileChooser : JFileChooser	-	Establece el valor del objeto grafico que importa un archivo.
getInterprete	-	Interprete	Obtiene el valor del inteprete.
setLabelExpresion	labelExpresion : JLabel	-	Establece el valor del label que tiene la imagen de la expresión de salida
getLabelExpresion	-	JLabel	Obtiene el valor del label que tiene la imagen de la expresión de salida

Mnemónico	*Clase*		
CP-4	PanelPrincipal		
verificarEstado	-	-	Verifica los estados de los documentos en el panel
cambiarEstadosPaneles	-	-	Cambia los estados de las áreas de texto dependiendo del estado de los documentos.
guardarArchivo	-	-	Genera y guarda un archivo
generarImagen	-	-	Genera la imagen de la expresión matemática
cambiarPath	opcion : int	String	Cambia el path obtenido por uno valido, según sea el caso: 1 para extensión tex 2 para extensión png

Tabla 11.4 Especificación Clase CP-4 PanelPrincipal

- Clase VentanaPrincipal: Esta clase se creo con el fin de contener todos los componentes gráficos del aplicativo. La estructura de la clase se presenta a continuación:

Mnemónico	*Clase*			
CP-5	VentanaPrincipal			
Atributos				
Nombre	*Tipo*	*Valor por defecto*	*Semántica*	*Visibilidad*
hs	HelpSet	null	Permite crear la ayuda de la aplicación (Ver especificación completa en el anexo 15.2.15)	Prívate

Mnemónico	*Clase*			
CP-5	VentanaPrincipal			
hb	HelpBroker	null	Permite cargar y visualizar la ayuda de la aplicación (Ver especificación completa en el anexo 15.2.14)	Prívate
tamañoVentana	Rectangle	null	Variable que establece el tamaño inicial de la ventana	Prívate
TabPanelCentral	JTabbedPane	null	Contenedor central de las pestañas.	Prívate
fileChooserBasico	JFileChooser	null	Objeto que me permite importar un archivo	Prívate
barraMenuUnico	JMenuBar	null	Barra de menú que contiene los menús de la aplicación.	Prívate
menuArchivo	JMenu	null	Menú que contiene las acciones de cerrar e importar.	Prívate
menuAyuda	JMenu	null	Menú que contiene las acciones de ayuda y acerca de.	Prívate
menuItemImportar	JMenuItem	null	Dispara la acción de importación de un archivo	Prívate
menuItemCerrar	JMenuItem	null	Dispara a acción de cierre de la aplicación.	Prívate
menuItemContenido	JMenuItem	null	Visualiza la ayuda.	Prívate

Mnemónico	***Clase***			
CP-5	VentanaPrincipal			
menuItemCreditos	JMenuItem	null	Muestra los datos de los creadores de la aplicación.	Prívate
validador	Validador	null	Objeto de negocio que realiza la validación de un archivo de entrada.	Prívate
documentoImportado	Documento	null	Objeto de negocio que representa un documento de entrada	Prívate
documentoTraducido	Documento	null	Objeto de negocio que representa un documento de salida.	Prívate
areaLocal	JTextArea	null	Contiene el texto plano de un archivo de entrada.	Prívate
vectorDocumentosImportados	Vector	null	Contiene el conjunto de archivos importados.	Prívate
panelPrincipalLocal	PanelPrincipal	null	Objeto grafico que se instancia cada vez que se importa un archivo.	Prívate
dialogoAcercaDe	DialogoAcercaDe	null	Cuadro de dialogo único que contiene los datos de los creadores de la aplicación.	Prívate
Métodos				
Nombre	***Parámetros (tipo)***		***Tipo de***	***Semántica***

Mnemónico	***Clase***		
CP-5	VentanaPrincipal		
		retorno	
inicializar	-	-	Inicializa los atributos de un objeto de tipo VentanaPrincipal
getTabPanelCentral	-	JTabbedPane	Obtiene el valor del objeto contendor de las pestañas.
getDialogoAcercaDe	-	DialogoAcercaDe	Obtiene el valor del cuadro de dialogo de créditos de la aplicación.
getDocumentoImportado	-	Documento	Obtiene el valor del documento importado.
setDocumentoImportado	documentoImportado : Documento	-	Establece el valor del documento importado.
getDocumentoTraducido	-	Documento	Obtiene el documento traducido.
setDocumentoTraducido	documentoTraducido : Documento	-	Establece el valor del documento traducido.
setTabPanelCentral	JTabbedPane	-	Establece el valor del contenedor de las pestañas.
getBarraMenuInicio	-	JMenuBar	Obtiene la barra de menú e inicializa sus eventos.
setBarraMenuUnico	barraMenuUnico : JMenuBar	-	Establece el valor de la barra de menú.
getMenuItemImportar	-	JMenuItem	Obtiene y establece el evento para el

Mnemónico	*Clase*		
CP-5	VentanaPrincipal		
			menú ítem de importar.
getMenuItemCerrar	-	JMenuItem	Obtiene y establece el evento para el menú ítem de cerrar la aplicación.
getMenuItemContenido	-	JMenuItem	Obtiene y establece el evento para el menú ítem de mostrar la ayuda de la aplicación.
getMenuItemCreditos	-	JMenuItem	Obtiene y establece el evento para el menú ítem de mostrar la información de los creadores de la aplicación.
getMenuArchivo	-	JMenu	Obtiene el valor del menú archivo.
getFileChooserBasico	-	JFileChooser	Obtiene el valor del objeto para importar un archivo.
setFileChooserBasico	fileChooserBasico: JFileChooser	-	Establece el valor del objeto que permite importar un archivo.
getValidador	-	Validador	Obtiene el valor del objeto validador
setValidador	Validador : Validador	-	Establece el valor del objeto validador
getVectorDocumentosImportados	-	Vector	Obtiene el vector de documentos

Mnemónico	*Clase*		
CP-5	VentanaPrincipal		
			importados.
setVectorDocumentos	vectorDocumentosImportados : Vector	-	Establece el valor para el vector de documentos importados.
getMenuAyuda	-	JMenu	Obtiene el valor del menú ayuda.
getPanelPrincipalLocal	-	PanelPrincipal	Obtiene el valor del objeto panel principal local por cada archivo importado.
setPanelPrincipalLocal	panelPrincipalLocal : PanelPrincipal	-	Establece el valor del objeto panel principal local.
actionPerformed	evento : ActionEvent	-	Controla las acciones realizadas por el menú.
validarExtension	file : File	boolean	Método para validar la extensión de un archivo importado
salir	-	-	Método para salir de la aplicación de manera general
verificarArchivosCargados	-	boolean	Método para Asegurar que un archivo solo se cargue una vez
cerrarTab	documentoBorrable : Documento, panelACerrar : PanelPrincipal	-	Cierra una pestaña con un documento referenciado y lo borra del vector de documentos.
cargarAyuda	-	-	Inicializa la ayuda de la

Mnemónico	*Clase*		
CP-5	VentanaPrincipal		
			aplicación.
ayuda	-	-	Visualiza la ayuda de la aplicación.

Tabla 11.5 Especificación Clase CP-5 VentanaPrincipal

- Clase Principal: Esta clase contiene el método *main* de la aplicación, por lo tanto tiene un único método, no tiene atributos y su código fuente puede ser consultado en documentación/anexos/ANEXO1 en el CD adjunto a este informe.

11.3.2 CAPA DE NEGOCIO

- Clase Constantes: Esta clase se creo con el fin de almacenar ciertas constantes para la aplicación que se invocan desde los diferentes procesos que esta realiza. Su estructura fue presentada en la sección 7.2.2 y su código fuente puede ser consultado documentación/anexos/ANEXO2 en el CD adjunto a este informe.

- Clase Documento: Esta clase se hizo con el fin de representar un documento de entrada o salida. Su estructura fue presentada en la sección 7.2.2 y su código fuente puede ser consultado en documentación/anexos/ANEXO1 en el CD adjunto a este informe.

- Clase Intérprete: Esta clase se creo con el fin de interpretar el código TeX y obtener una imagen con los símbolos matemáticos de salida. Su estructura fue presentada en la sección 7.2.2 y su código fuente puede ser consultado en documentación/anexos/ANEXO1 en el CD adjunto a este informe.

- Clase Traductor: Esta clase se creo con el propósito de realizar el proceso de transformación entre código XML y TeX. Su estructura fue presentada en la sección 7.2.2 y su código fuente puede ser consultado en documentación/anexos/ANEXO1 en el CD adjunto a este informe.

- Clase Validador: Esta clase se creo con el fin de realizar el proceso de validación de un documento XML de entrada con respecto al esquema transformerUmlarchro.xsd (Sección 7.3.1). Su estructura fue presentada en la sección 7.2.2 y su código fuente puede ser consultado en documentación/anexos/ANEXO1 en el CD adjunto a este informe.

11.3.3 CAPA DE PERSISTENCIA

Esta capa está representada por aquellos archivos que utiliza la aplicación desarrollada para la traducción de un documento XML a la sintaxis del cálculo – ρ_{arq}. Entre estos archivos encontramos el esquema transformerUmlarchro.xsd y la hoja de estilos transformerumlarchroxsl.xsl, su estructura y código fuente pueden ser consultados de la siguiente forma:

- Estructura del esquema transformerUmlarchro.xsd: documentación/anexos/ANEXO5 en el CD adjunto a este informe.
- Código fuente del esquema transformerUmlarchro.xsd: documentación/anexos/ANEXO3 en el CD adjunto a este informe.
- Estructura de la hoja de estilos transformerumlarchroxsl.xsl: documentación/anexos/ANEXO4 en el CD adjunto a este informe.
- Código Fuente de la hoja de estilos transformerumlarchroxsl.xsl: documentación/anexos/ANEXO3 en el CD adjunto a este informe.

La descripción se encuentra en las secciones 7.3.1 (Esquema transformerUmlarchro.xsd) y 7.3.2 (Hoja de estilos transformerumlarchroxsl.xsl).

12 EJECUCIÓN DE PRUEBAS FUNCIONALES

En este capítulo se describirá el proceso de validación funcional de la aplicación desarrollada, describiendo la justificación de éstas pruebas y cada uno de los casos diseñados para la respectiva verificación.

12.1 JUSTIFICACIÓN DE LAS PRUEBAS

Cuando un programa es implementado para proveer una representación concreta de un algoritmo, los desarrolladores de este programa están naturalmente preocupados por la correctitud y funcionamiento de la implementación. La verificación de software es el proceso de garantizar que un programa encuentre su adecuada especificación [Tucker, 2004], y con el fin de realizar este proceso se construyeron cinco casos de prueba que serán descritos a continuación.

12.2 EJECUCIÓN DE LAS PRUEBAS

Con el fin de verificar las funcionalidades de la aplicación se diseñaron 5 casos de prueba con los cuales se busca cubrir todas las variables que debe contemplar la aplicación desarrollada.

12.2.1 Caso 1

Objetivo: Evaluar la transformación de un documento XML de entrada que describe la siguiente arquitectura de software:

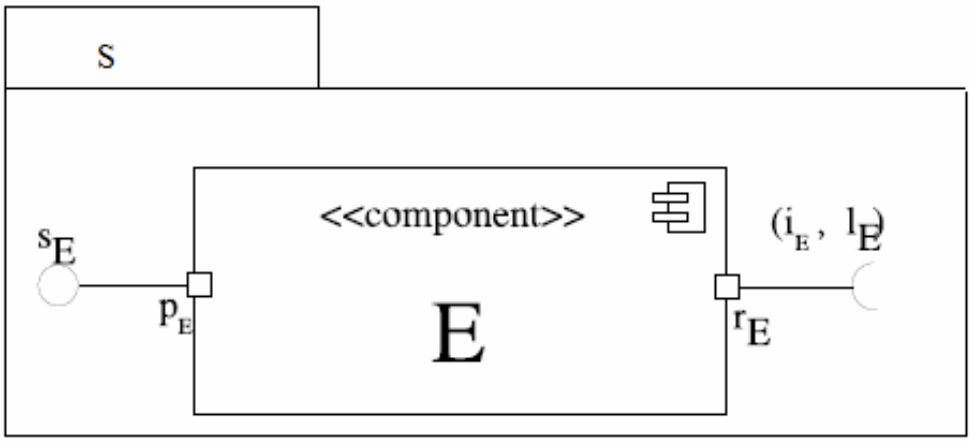

Figura 12.1 Descripción Gráfica Arquitectura Un Componente

Código XML del archivo de entrada:

```
<?xml version="1.0" encoding="UTF-8"?>
<archinstance:xArch xmlns:archinstance="instance.xsd"
xmlns:archtypes="types.xsd" xmlns:boolguard="boolguard.xsd"
xmlns:umlstructure="umlarchro.xsd"
xmlns:transforUmlstructure="transformerUmlarchro.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="transformerUmlarchro.xsd transformerUmlarchro.xsd">
    <transforUmlstructure:transforArchRo transforUmlstructure:id="S">
        <transforUmlstructure:description>S</transforUmlstructure:description>
        <transforUmlstructure:transformComponentRo archtypes:id="E">
            <archtypes:description>E</archtypes:description>
            <umlstructure:portRo archtypes:id="p1">
                <archtypes:description>p1</archtypes:description>
                <umlstructure:interfaceRo archtypes:id="pi1">
                    <archtypes:description>pi1</archtypes:description>
```

```
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>E</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>p1</umlstructure:parameter>
                <umlstructure:subindex>E</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="r1">
                <archtypes:description>r1</archtypes:description>
                <umlstructure:interfaceRo archtypes:id="ri1">
                    <archtypes:description>ri1</archtypes:description>
                    <umlstructure:umlType>required</umlstructure:umlType>
                    <umlstructure:parameter>i</umlstructure:parameter>
                    <umlstructure:parameter>l</umlstructure:parameter>
                    <umlstructure:subindex>E</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>r1</umlstructure:parameter>
                <umlstructure:subindex>E</umlstructure:subindex>
                <umlstructure:umlType>required</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>initial</transforUmlstructure:rolRo>
        </transforUmlstructure:transformComponentRo>
    </transforUmlstructure:transforArchRo>
</archinstance:xArch>
```

Conjunto de símbolos de salida esperados:

$$\mathbf{S} \stackrel{def}{=} p_E : x/xs_E \wedge r_E :: y/yl_E \wedge l_E :: i_E/E$$

12.2.2 Caso 2

Objetivo: Evaluar la transformación de un documento XML de entrada que describe la siguiente arquitectura de software:

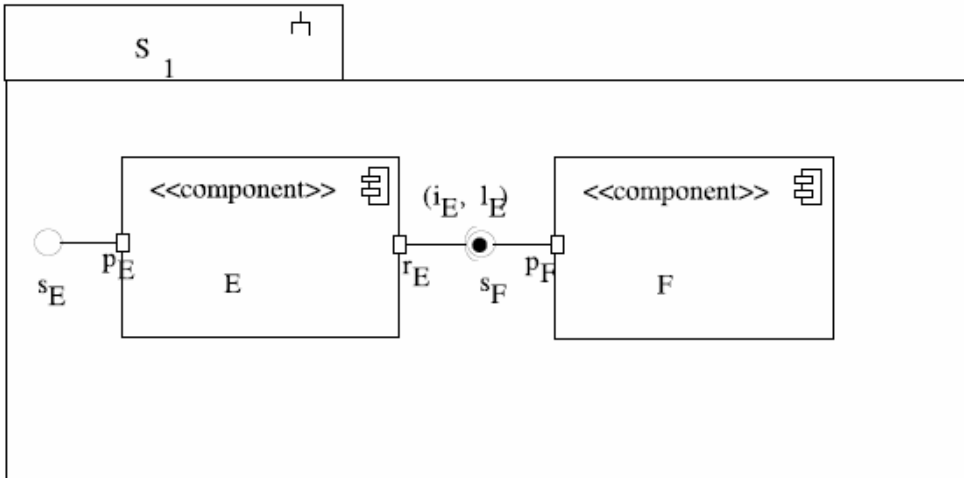

Figura 12.2 Descripción Gráfica Arquitectura Dos Componentes

Código XML del archivo de entrada:

```
<?xml version="1.0" encoding="UTF-8"?>
<archinstance:xArch xmlns:archinstance="instance.xsd"
xmlns:archtypes="types.xsd" xmlns:boolguard="boolguard.xsd"
xmlns:umlstructure="umlarchro.xsd"
xmlns:transforUmlstructure="transformerUmlarchro.xsd"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="transformerUmlarchro.xsd transformerUmlarchro.xsd">
    <transforUmlstructure:transforArchRo transforUmlstructure:id="S2">

    <transforUmlstructure:description>S2</transforUmlstructure:description>
        <transforUmlstructure:transformComponentRo archtypes:id="E">
            <archtypes:description>E</archtypes:description>
            <umlstructure:portRo archtypes:id="pe1">
                <archtypes:description>puerto provision</archtypes:description>
                <umlstructure:interfaceRo archtypes:id="pe1.1">
                    <archtypes:description>Interface de provision </ archtypes
                        :description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>E</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pe1</umlstructure:parameter>
                <umlstructure:subindex>E</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
```

```
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="re1">
                <archtypes:description>Puerto de requerimiento </ archtypes :
                description>
                <umlstructure:interfaceRo archtypes:id="re1.1">
                    <archtypes:description>Interfaz de requerimiento
                    </archtypes: description>
                    <umlstructure:umlType>required</umlstructure:umlType>
                    <umlstructure:parameter>i</umlstructure:parameter>
                    <umlstructure:parameter>l</umlstructure:parameter>
                    <umlstructure:subindex>E</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>re1</umlstructure:parameter>
                <umlstructure:subindex>E</umlstructure:subindex>
                <umlstructure:umlType>required</umlstructure:umlType>
            </umlstructure:portRo>
        <transforUmlstructure:rolRo>intermediate</transforUmlstructure:rolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:transformComponentRo archtypes:id="F">
            <archtypes:description>F</archtypes:description>
            <umlstructure:portRo archtypes:id="pf1">
                <archtypes:description>Puerto de provision 1 </archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pf1.1">
                    <archtypes:description>Interfaz de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>F</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pf1</umlstructure:parameter>
                <umlstructure:subindex>F</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>initial</transforUmlstructure:rolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:connectorRo archinstance:id="C1">
            <archinstance:description>Conector entre E y F</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="re1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
```

```
                    <archinstance:anchorOnInterface xlink:href="pf1" xlink:type=
                    "simple"/>
                </archinstance:point>
            </transforUmlstructure:connectorRo>
        </transforUmlstructure:transforArchRo>
</archinstance:xArch>
```

Conjunto de símbolos de salida esperados:

$$S_1 = \{(p_E : x/xs_E) \wedge (r_E :: y/yl_E) \wedge (l_E :: i_E/E)\} \wedge \{(p_F : z/zs_F)\} \wedge \{r_E p_F\}$$

12.2.3 Caso 3

Objetivo: Evaluar la transformación de un documento XML de entrada que describe la siguiente arquitectura de software:

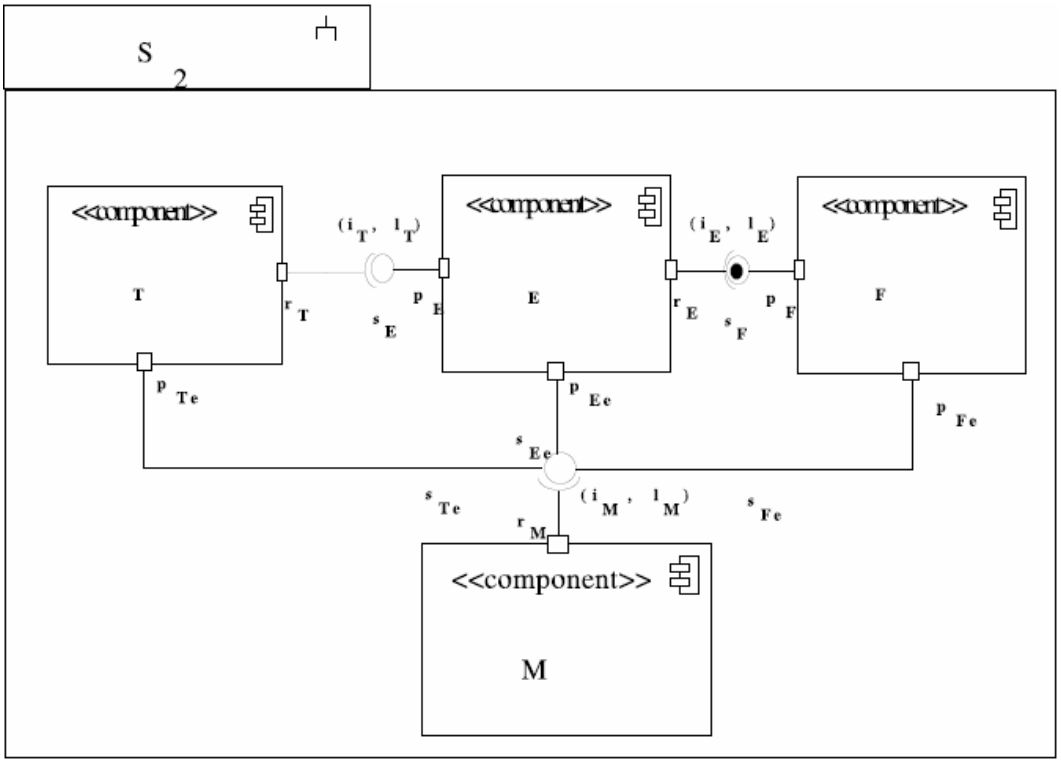

Figura 12.3 Gráfica Arquitectura Cuatro Componentes

Codigo XML del archivo de entrada:

```
<?xml version="1.0" encoding="UTF-8"?>
<archinstance:xArch xmlns:archinstance="instance.xsd"
xmlns:archtypes="types.xsd" xmlns:boolguard="boolguard.xsd"
xmlns:umlstructure="umlarchro.xsd"
xmlns:transforUmlstructure="transformerUmlarchro.xsd"
xmlns:options="options.xsd" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="transformerUmlarchro.xsd transformerUmlarchro.xsd">

    <transforUmlstructure:transforArchRo transforUmlstructure:id="S3">

    <transforUmlstructure:description>S3</transforUmlstructure:description>
        <transforUmlstructure:transformComponentRo archtypes:id="F">
            <archtypes:description>F</archtypes:description>
            <umlstructure:portRo archtypes:id="pf1">
                <archtypes:description>puerto provision</archtypes:description>
                <umlstructure:interfaceRo archtypes:id="pf1.1">
                    <archtypes:description>Interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>F</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pf1</umlstructure:parameter>
                <umlstructure:subindex>F</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="pfe">
                <archtypes:description>puerto de provision</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pfe.1">
                    <archtypes:description>interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>Fe</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pfe</umlstructure:parameter>
```

```
                <umlstructure:subindex>Fe</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>initial</transforUmlstructure:rolRo>
            <transforUmlstructure:expBoolRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="E">
            <transforUmlstructure:estado>true</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="M">
            <transforUmlstructure:estado>false</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
            </transforUmlstructure:expBoolRo>
        </transforUmlstructure:transformComponentRo>
    <transforUmlstructure:transformComponentRo archtypes:id="E">
            <archtypes:description>E</archtypes:description>
            <umlstructure:portRo archtypes:id="re1">
                <archtypes:description>puerto de requerimiento</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="re1.1">
                    <archtypes:description>interface de requerimiento
                    </archtypes:
                    description>
                    <umlstructure:umlType>required</umlstructure:umlType>
                    <umlstructure:parameter>i</umlstructure:parameter>
                    <umlstructure:parameter>l</umlstructure:parameter>
                    <umlstructure:subindex>E</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>re1</umlstructure:parameter>
                <umlstructure:subindex>E</umlstructure:subindex>
                <umlstructure:umlType>required</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="pe1">
                <archtypes:description>puerto de provision
                </archtypes:description>
                <umlstructure:interfaceRo archtypes:id="pe1.1">
                    <archtypes:description>interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>E</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pe1</umlstructure:parameter>
```

```
                <umlstructure:subindex>E</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="pee">
                <archtypes:description>puerto de provision</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pee.1">
                    <archtypes:description>interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>Ee</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pee</umlstructure:parameter>
                <umlstructure:subindex>Ee</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
        <transforUmlstructure:rolRo>intermediate</transforUmlstructure:rolRo>
            <transforUmlstructure:expBoolRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="T">
            <transforUmlstructure:estado>true</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="M">

    <transforUmlstructure:estado>false</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
            </transforUmlstructure:expBoolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:transformComponentRo archtypes:id="T">
            <archtypes:description>T</archtypes:description>
            <umlstructure:portRo archtypes:id="rt1">
                <archtypes:description>puerto de requerimiento</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="rt1.1">
                    <archtypes:description>interface de requerimiento
                    </archtypes:
                    description>
                    <umlstructure:umlType>required</umlstructure:umlType>
                    <umlstructure:parameter>i</umlstructure:parameter>
                    <umlstructure:parameter>l</umlstructure:parameter>
                    <umlstructure:subindex>T</umlstructure:subindex>
                </umlstructure:interfaceRo>
```

```
                <umlstructure:parameter>rt1</umlstructure:parameter>
                <umlstructure:subindex>T</umlstructure:subindex>
                <umlstructure:umlType>required</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="pte">
                <archtypes:description>puerto de provision</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pte.1">
                    <archtypes:description>interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>Te</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pte</umlstructure:parameter>
                <umlstructure:subindex>Te</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>final</transforUmlstructure:rolRo>
            <transforUmlstructure:expBoolRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="S3">

<transforUmlstructure:estado>true</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="M">
            <transforUmlstructure:estado>false</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
                </transforUmlstructure:referenceRo>
            </transforUmlstructure:expBoolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:transformComponentRo archtypes:id="M">
            <archtypes:description>M</archtypes:description>
            <umlstructure:portRo archtypes:id="rm1">
                <archtypes:description>puerto de requerimiento</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="rm1.1">
                    <archtypes:description>interface de requerimiento
                    </archtypes:
                    description>
                    <umlstructure:umlType>required</umlstructure:umlType>
                    <umlstructure:parameter>i</umlstructure:parameter>
                    <umlstructure:parameter>l</umlstructure:parameter>
```

```
                    <umlstructure:subindex>M</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>rm1</umlstructure:parameter>
                <umlstructure:subindex>M</umlstructure:subindex>
                <umlstructure:umlType>required</umlstructure:umlType>
            </umlstructure:portRo>
<transforUmlstructure:rolRo>errorHandler</transforUmlstructure:rolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:connectorRo archinstance:id="C1">
            <archinstance:description>Conector entre E y F</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="re1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pf1" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
        <transforUmlstructure:connectorRo archinstance:id="C2">
            <archinstance:description>Conector entre T y E</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="rt1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pe1" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
        <transforUmlstructure:connectorRo archinstance:id="Cm1">
            <archinstance:description>Conector entre M y F</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="rm1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pfe" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
```

```
        <transforUmlstructure:connectorRo archinstance:id="Cm2">
            <archinstance:description>Conector entre M y E</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="rm1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pee" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
        <transforUmlstructure:connectorRo archinstance:id="Cm3">
            <archinstance:description>Conector entre M y T</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="rm1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pte" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
    </transforUmlstructure:transforArchRo>
</archinstance:xArch>
```

Conjunto de símbolos de salida esperados:

$$S_2 = \{if\ \underline{F}^{\top}\ then\ (p_F : z/zs_F)\ \wedge\ (p_{Fe} : w/ws_{Fe}) \wedge\ C_{FE}\ \wedge$$
$$(p_E : x/xs_E)\ \wedge\ (r_E :: y/yl_E)\ \wedge\ (p_{Ee} : v/vs_{Ee})\ \wedge\ (l_E :: i_E/E)\ \ else\ C_{FM}\ \wedge$$
$$(r_M : y/yl_M)\ \wedge\ (l_M : i_M/M)\}\ \wedge\ \{if\ E^{\top}\ then\ C_{ET}\ \wedge\ (r_T :: q/ql_T)\ \wedge$$
$$(l_T : i_T/T)\ \wedge\ (p_{Te} : n/ns_{Te})\ else\ C_{EM}\ \wedge\ (r_M : y/yl_M)\ \wedge\ (l_M : i_M/M)\}\ \wedge$$
$$\{if\ T^{\top}\ then\ S_2\ =\ \acute{e}xito\ else\ C_{TM}\ \wedge (r_M : y/yl_M)\ \wedge\ (l_M : i_M/M)\}$$

12.2.4 Caso 4

Objetivo: Evaluar la transformación de un documento XML de entrada que describe la siguiente arquitectura de software:

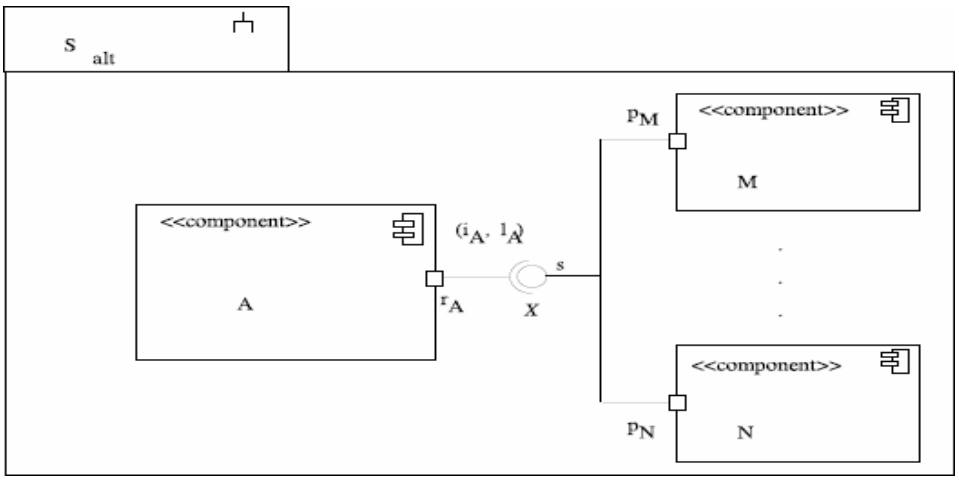

Figura 12.4 Gráfica Arquitectura Componentes Opcionales

Codigo XML del archivo de entrada:

```
<?xml version="1.0" encoding="UTF-8"?>
<archinstance:xArch xmlns:archinstance="instance.xsd"
xmlns:archtypes="types.xsd" xmlns:boolguard="boolguard.xsd"
xmlns:umlstructure="umlarchro.xsd"
xmlns:transforUmlstructure="transformerUmlarchro.xsd"
xmlns:options="options.xsd" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="transformerUmlarchro.xsd transformerUmlarchro.xsd">

    <transforUmlstructure:transforArchRo transforUmlstructure:id="Salt">

    <transforUmlstructure:description>Salt</transforUmlstructure:description>
        <transforUmlstructure:transformComponentRo archtypes:id="A">
            <archtypes:description>A</archtypes:description>
            <umlstructure:portRo archtypes:id="ra1">
```

```
                <archtypes:description>Puerto de requerimiento</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="ra1.1">
                    <archtypes:description>Interfaz de requerimiento
                    </archtypes:
                    description>
                    <umlstructure:umlType>required</umlstructure:umlType>
                    <umlstructure:parameter>i</umlstructure:parameter>
                    <umlstructure:parameter>l</umlstructure:parameter>
                    <umlstructure:subindex>A</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>ra1</umlstructure:parameter>
                <umlstructure:subindex>A</umlstructure:subindex>
                <umlstructure:umlType>required</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>final</transforUmlstructure:rolRo>
            <transforUmlstructure:variableSet>

<transforUmlstructure:variable>X</transforUmlstructure:variable>
            </transforUmlstructure:variableSet>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:transformComponentRo archtypes:id="M">
            <archtypes:description>M</archtypes:description>
            <umlstructure:portRo archtypes:id="pm1">
                <archtypes:description>Puerto de provision</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pm1.1">
                    <archtypes:description>Interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>s</umlstructure:parameter>
                    <umlstructure:subindex>M</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pm1</umlstructure:parameter>
                <umlstructure:subindex>M</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>option</transforUmlstructure:rolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:transformComponentRo archtypes:id="N">
            <archtypes:description>N</archtypes:description>
            <umlstructure:portRo archtypes:id="pn1">
                <archtypes:description>Puerto de provision</archtypes:
                description>
```

```
                <umlstructure:interfaceRo archtypes:id="pn1.1">
                    <archtypes:description>Interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>s</umlstructure:parameter>
                    <umlstructure:subindex>N</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pn1</umlstructure:parameter>
                <umlstructure:subindex>N</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>option</transforUmlstructure:rolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:optionalConnectorRo archinstance:id="CO1">
            <archinstance:description>Conector Opcional entre M y A</
            archinstance:description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="ra1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pm1" xlink:type=
                "simple"/>
            </archinstance:point>
            <transforUmlstructure:boolGuard>
                <booleanExp>
                    <equals>
                        <symbol>X</symbol>
                        <symbol2>pm1</symbol2>
                    </equals>
                </booleanExp>
            </transforUmlstructure:boolGuard>
        </transforUmlstructure:optionalConnectorRo>
        <transforUmlstructure:optionalConnectorRo archinstance:id="CO2">
            <archinstance:description>Conector Opcional entre N y A</
            archinstance:description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="ra1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pn1" xlink:type=
                "simple"/>
            </archinstance:point>
```

```
            <transforUmlstructure:boolGuard>
                <booleanExp>
                    <equals>
                        <symbol>X</symbol>
                        <symbol2>pn1</symbol2>
                    </equals>
                </booleanExp>
            </transforUmlstructure:boolGuard>
        </transforUmlstructure:optionalConnectorRo>
    </transforUmlstructure:transforArchRo>
</archinstance:xArch>
```

Conjunto de símbolos de salida esperados:

$$S_{alt} \stackrel{def}{=} [if\ \exists X((X = p_M)\ then\ \{if\ M^{\top}\ then\ r_A :: y/yl_A\ \wedge\ l_A :: i_A/A\ \wedge\ r_A p_M \wedge p_M : x/xs\})\ \cdots\ ((X = p_N)\ then\ \{if\ N^{\top}\ then\ r_A :: y/yl_A\ \wedge\ l_A :: i_A/A\ \wedge\ r_A p_N \wedge p_N : x/xs\})\ else\ Manejarerror]$$

12.2.5 Caso 5

Objetivo: Evaluar la transformación de un documento XML de entrada que describe la siguiente arquitectura de software:

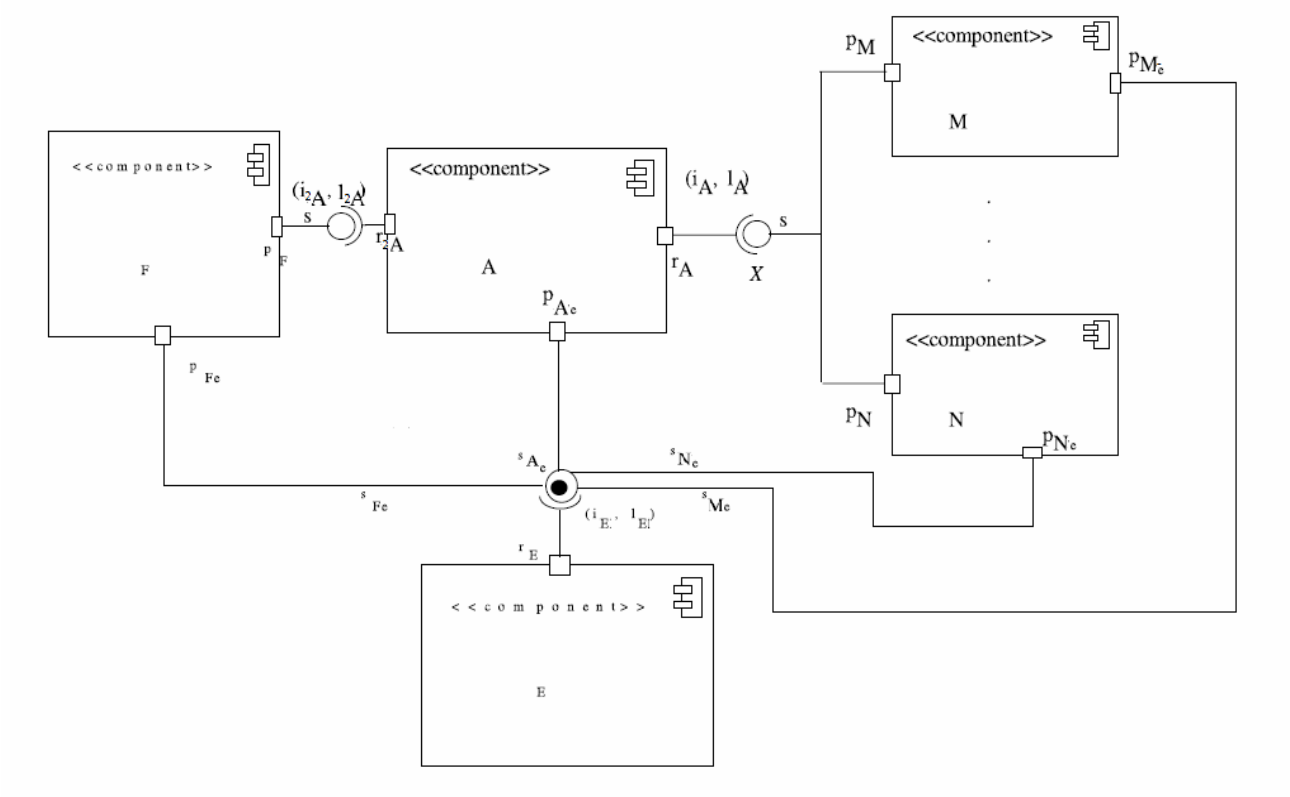

Figura 12.5 Gráfica Arquitectura Combinada

La arquitectura planteada combina componentes opcionales, componentes sencillos y manejo de error. Se presentara el código de entrada XML, pero se obviara el conjunto de símbolos esperados dado que en Diosa, 2005] no existe una arquitectura similar a la aquí presentada. También se aclara que esta arquitectura de prueba es una modificación del caso de prueba 4.

Codigo XML del archivo de entrada:

```
<?xml version="1.0" encoding="UTF-8"?>
<archinstance:xArch xmlns:archinstance="instance.xsd"
xmlns:archtypes="types.xsd" xmlns:boolguard="boolguard.xsd"

xmlns:umlstructure="umlarchro.xsd"
xmlns:transforUmlstructure="transformerUmlarchro.xsd"
```

```
xmlns:options="options.xsd" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="transformerUmlarchro.xsd transformerUmlarchro.xsd">
    <transforUmlstructure:transforArchRo transforUmlstructure:id="Salt">

    <transforUmlstructure:description>Salt</transforUmlstructure:description>
        <transforUmlstructure:transformComponentRo archtypes:id="A">
            <archtypes:description>A</archtypes:description>
            <umlstructure:portRo archtypes:id="ra1">
                <archtypes:description>Puerto de requerimiento</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="ra1.1">
                    <archtypes:description>Interfaz de requerimiento</archtypes:
                    description>
                    <umlstructure:umlType>required</umlstructure:umlType>
                    <umlstructure:parameter>i</umlstructure:parameter>
                    <umlstructure:parameter>l</umlstructure:parameter>
                    <umlstructure:subindex>A</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>ra1</umlstructure:parameter>
                <umlstructure:subindex>A</umlstructure:subindex>
                <umlstructure:umlType>required</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="ra2">
                <archtypes:description>Puerto de requerimiento</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="ra2.1">
                    <archtypes:description>Interfaz de requerimiento</archtypes:
                    description>
                    <umlstructure:umlType>required</umlstructure:umlType>
                    <umlstructure:parameter>i</umlstructure:parameter>
                    <umlstructure:parameter>l</umlstructure:parameter>
                    <umlstructure:subindex>A</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>ra2</umlstructure:parameter>
                <umlstructure:subindex>A</umlstructure:subindex>
                <umlstructure:umlType>required</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="pae">
                <archtypes:description>puerto de provision</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pae.1">
                    <archtypes:description>interface de provision</archtypes:
                    description>
```

```
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>s</umlstructure:parameter>
                    <umlstructure:subindex>Ae</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pae</umlstructure:parameter>
                <umlstructure:subindex>Ae</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>final</transforUmlstructure:rolRo>
            <transforUmlstructure:expBoolRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="Salt">
            <transforUmlstructure:estado>true</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="E">
            <transforUmlstructure:estado>false</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
            </transforUmlstructure:expBoolRo>
            <transforUmlstructure:variableSet>

<transforUmlstructure:variable>X</transforUmlstructure:variable>
            </transforUmlstructure:variableSet>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:transformComponentRo archtypes:id="M">
            <archtypes:description>M</archtypes:description>
            <umlstructure:portRo archtypes:id="pm1">
                <archtypes:description>Puerto de provision</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pm1.1">
                    <archtypes:description>Interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>s</umlstructure:parameter>
                    <umlstructure:subindex>M</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pm1</umlstructure:parameter>
                <umlstructure:subindex>M</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="pme">
                <archtypes:description>puerto de provision</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pme.1">
```

```
                    <archtypes:description>interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>s</umlstructure:parameter>
                    <umlstructure:subindex>Me</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pme</umlstructure:parameter>
                <umlstructure:subindex>Me</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>option</transforUmlstructure:rolRo>
            <transforUmlstructure:expBoolRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="A">
            <transforUmlstructure:estado>true</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="E">
            <transforUmlstructure:estado>false</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
            </transforUmlstructure:expBoolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:transformComponentRo archtypes:id="N">
            <archtypes:description>N</archtypes:description>
            <umlstructure:portRo archtypes:id="pn1">
                <archtypes:description>Puerto de provision</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pn1.1">
                    <archtypes:description>Interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>N</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pn1</umlstructure:parameter>
                <umlstructure:subindex>N</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="pne">
                <archtypes:description>puerto de provision</archtypes
                :description>
                <umlstructure:interfaceRo archtypes:id="pne.1">
                    <archtypes:description>interface de provision</archtypes:
                    description>
```

```
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>s</umlstructure:parameter>
                    <umlstructure:subindex>Ne</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pne</umlstructure:parameter>
                <umlstructure:subindex>Ne</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>option</transforUmlstructure:rolRo>
            <transforUmlstructure:expBoolRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="A">
<transforUmlstructure:estado>true</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="E">
<transforUmlstructure:estado>false</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
            </transforUmlstructure:expBoolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:transformComponentRo archtypes:id="E">
            <archtypes:description>E</archtypes:description>
            <umlstructure:portRo archtypes:id="re1">
                <archtypes:description>puerto de requerimiento</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="re1.1">
                    <archtypes:description>interface de requerimiento</archtypes:
                    description>
                    <umlstructure:umlType>required</umlstructure:umlType>
                    <umlstructure:parameter>i</umlstructure:parameter>
                    <umlstructure:parameter>l</umlstructure:parameter>
                    <umlstructure:subindex>E</umlstructure:subindex>
                </umlstructure:interfaceRo>
<umlstructure:parameter>re1</umlstructure:parameter>
                <umlstructure:subindex>E</umlstructure:subindex>
                <umlstructure:umlType>required</umlstructure:umlType>
            </umlstructure:portRo>
<transforUmlstructure:rolRo>errorHandler</transforUmlstructure:rolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:transformComponentRo archtypes:id="F">
            <archtypes:description>F</archtypes:description>
            <umlstructure:portRo archtypes:id="pf1">
                <archtypes:description>Puerto de provision 1</archtypes:
                description>
```

```
                <umlstructure:interfaceRo archtypes:id="pf1.1">
                    <archtypes:description>Interfaz de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>s</umlstructure:parameter>
                    <umlstructure:subindex>F</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pf1</umlstructure:parameter>
                <umlstructure:subindex>F</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="pfe">
                <archtypes:description>puerto de provision</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pfe.1">
                    <archtypes:description>interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>s</umlstructure:parameter>
                    <umlstructure:subindex>Fe</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pfe</umlstructure:parameter>
                <umlstructure:subindex>Fe</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>initial</transforUmlstructure:rolRo>
            <transforUmlstructure:expBoolRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="A">
<transforUmlstructure:estado>true</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
                <transforUmlstructure:referenceRo transforUmlstructure:
                referenciaRo="E">
<transforUmlstructure:estado>false</transforUmlstructure:estado>
                </transforUmlstructure:referenceRo>
            </transforUmlstructure:expBoolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:connectorRo archinstance:id="Caf">
            <archinstance:description>Conector entre A y F</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="ra2" xlink:type=
                "simple"/>
            </archinstance:point>
```

```
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pf1" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
        <transforUmlstructure:connectorRo archinstance:id="Ce4">
            <archinstance:description>Conector entre E y F</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="re1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pfe" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
        <transforUmlstructure:connectorRo archinstance:id="Ce1">
            <archinstance:description>Conector entre E y A</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="re1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pae" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
        <transforUmlstructure:connectorRo archinstance:id="Ce2">
            <archinstance:description>Conector entre E y M</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="re1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pme" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
        <transforUmlstructure:connectorRo archinstance:id="Ce3">
            <archinstance:description>Conector entre E y N</archinstance:
            description><archinstance:point>
```

```
                <archinstance:anchorOnInterface xlink:href="re1" xlink:type=
                "simple"/></archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pne" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
        <transforUmlstructure:optionalConnectorRo archinstance:id="CO1">
            <archinstance:description>Conector Opcional entre M y A</
            archinstance:description><archinstance:point>
                <archinstance:anchorOnInterface xlink:href="ra1" xlink:type=
                "simple"/></archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pm1" xlink:type=
                "simple"/></archinstance:point>
            <transforUmlstructure:boolGuard>
                <booleanExp>
                    <equals>
                        <symbol>X</symbol>
                        <symbol2>pm1</symbol2>
                    </equals>
                </booleanExp>
            </transforUmlstructure:boolGuard>
        </transforUmlstructure:optionalConnectorRo>
        <transforUmlstructure:optionalConnectorRo archinstance:id="CO2">
            <archinstance:description>Conector Opcional entre N y A</
            archinstance:description><archinstance:point>
                <archinstance:anchorOnInterface xlink:href="ra1" xlink:type=
                "simple"/></archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pn1" xlink:type=
                "simple"/></archinstance:point>
            <transforUmlstructure:boolGuard>
                <booleanExp>
                    <equals>
                        <symbol>X</symbol>
                        <symbol2>pn1</symbol2>
                    </equals>
                </booleanExp>
            </transforUmlstructure:boolGuard>
        </transforUmlstructure:optionalConnectorRo>
    </transforUmlstructure:transforArchRo>
</archinstance:xArch>
```

12.3 ANÁLISIS DE RESULTADOS DE LAS PRUEBAS

Al realizar la ejecución de las pruebas, se realizó la comparación entre el conjunto de símbolos esperados y el conjunto de símbolos obtenidos. Específicamente se contó la cantidad de símbolos resultantes de la transformación original y se comparó con respecto a la cantidad de símbolos resultantes de la transformación realizada con el prototipo desarrollado, utilizando la hoja de estilos diseñada; finalmente se computó mediante una regla de tres. Así pues los resultados fueron los siguientes:

12.3.1 Caso 1:

- Símbolos esperados:

$$\mathrm{S} \stackrel{def}{=} p_E : x/xs_E \wedge r_E :: y/yl_E \wedge l_E :: i_E/E$$

- Símbolos obtenidos:

$$S = \{p1_E : x/xS_E \wedge r1_E :: y/yl_E \wedge l_E :: i_E/E\}$$

- Porcentaje concordancia: 23 simbolos esperados a concordar, 19 simbolos correctos, concordancia 82 %
- Conclusión: La aplicación arrojó claramente la estructura de la arquitectura ingresada, sin embargo, agregó llaves de apertura y cierre por el componente E. El símbolo de equivalencia es distinto.

12.3.2 Caso 2:

- Símbolos esperados:

$$S_1 = \{(p_E : x/xs_E) \wedge (r_E :: y/yl_E) \wedge (l_E :: i_E/E)\} \wedge \{(p_F : z/zs_F)\} \wedge \{r_E p_F\}$$

- Símbolos obtenidos:

$$S2 = \{pe1_E : x/xS_E \wedge re1_E :: y/yl_E \wedge l_E :: i_E/E\} \wedge \{pf1_F : x/xS_F\} \wedge \{re1_E pf1_F\}$$

- Porcentaje concordancia: 45 Simbolos esperados a concordar, 37 simbolos correctos, concordancia 83 %
- Conclusión: La aplicación arrojó claramente la estructura de la arquitectura ingresada, pero obvió los paréntesis de separación entre puertos.

12.3.3 Caso 3:

- Símbolos esperados:

$$\begin{aligned} S_2 = \{&if\ \underline{F}^{\top}\ then\ (p_F : z/zs_F)\ \wedge\ (p_{Fe} : w/ws_{Fe}) \wedge C_{FE}\ \wedge \\ &(p_E : x/xs_E)\ \wedge\ (r_E :: y/yl_E)\ \wedge\ (p_{Ee} : v/vs_{Ee})\ \wedge\ (l_E :: i_E/E)\ \ else\ C_{FM}\ \wedge \\ &(r_M : y/yl_M)\ \wedge\ (l_M : i_M/M)\}\ \wedge\ \{if\ E^{\top}\ then\ C_{ET}\ \wedge (r_T :: q/ql_T)\ \wedge \\ &(l_T : i_T/T)\ \wedge\ (p_{Te} : n/ns_{Te})\ else\ C_{EM}\ \wedge (r_M : y/yl_M)\ \wedge\ (l_M : i_M/M)\}\ \wedge \\ &\{if\ T^{\top}\ then\ S_2\ =\ éxito\ else\ C_{TM}\ \wedge (r_M : y/yl_M)\ \wedge\ (l_M : i_M/M)\} \end{aligned}$$

- Símbolos obtenidos:

$$\begin{aligned} S3 = \{&if(F^{\top})then(pf1_F : x/xS_F \wedge pfe_Fe : x/xS_Fe \wedge re1_E pf1_F \wedge re1_E :: y/yl_E \wedge l_E :: i_E/E \\ &\wedge pe1_E : x/xS_E \wedge pee_Ee : x/xS_Ee)else(rm1_M pfe_Fe \wedge rm1_M :: y/yl_M \wedge l_M :: i_M/M)\} \\ &\wedge \{if(E^{\top})then(re1_E :: y/yl_E \wedge l_E :: i_E/E \wedge pe1_E : x/xS_E \wedge pee_Ee : x/xS_Ee \wedge rt1_T pe1_E \\ &\wedge rt1_T :: y/yl_T \wedge l_T :: i_T/T \wedge pte_Te : x/xS_Te)else(rm1_M pee_Ee \wedge rm1_M :: y/yl_M \wedge l_M :: i_M/M)\} \\ &\wedge \{if(T^{\top})then(S3 = EXITO)else(rm1_M pte_Te \wedge rm1_M :: y/yl_M \wedge l_M :: i_M/M)\} \end{aligned}$$

- Porcentaje concordancia: Simbolos esperados a concordar 156, simbolos correctos 135, concordancia 86 %
- Conclusión: La aplicación arrojó la estructura de la arquitectura ingresada con las siguientes diferencias:
- Agregó paréntesis de separación entre las expresiones booleanas.
- Obvió los paréntesis de separación entre puertos y conectores.

- Agregó la especificación de los conectores.

12.3.4 Caso 4:

- Símbolos esperados:

$$S_{alt} \stackrel{def}{=} [if\ \exists X((X = p_M)\ then\ \{if\ M^{\top}\ then\ r_A :: y/yl_A\ \wedge\ l_A :: i_A/A\ \wedge\ r_A p_M \wedge p_M : x/xs\})\ \cdots\ ((X = p_N)\ then\ \{if\ N^{\top}\ then\ r_A :: y/yl_A\ \wedge\ l_A :: i_A/A\ \wedge\ r_A p_N \wedge p_N : x/xs\})\ else\ Manejarerror]$$

- Símbolos obtenidos:

$$Salt = [if \exists X((X = pm1) then \{if(M^{\top}) then(ra1_A :: y/yl_A \wedge l_A :: i_A/A \wedge pm1_M : x/xS_M \wedge ra1_A pm1_M)\ else(Maneja...Error)\})\ldots((X = pn1) then \{if(N^{\top}) then(ra1_A :: y/yl_A \wedge l_A :: i_A/A \wedge pn1_N : x/xS_N \wedge ra1_A pn1_N) else(Maneja...Error)\}) else(Maneja...Error)]$$

- Porcentaje concordancia: Simbolos esperados a concordar 72, simbolos correctos 64, concordancia 88%
- Conclusión: La arquitectura arrojada por la aplicación es similar a la ingresada con las siguientes diferencias:

  - El símbolo de equivalencia es distinto.
  - Agregó paréntesis de separación entre las expresiones booleanas.
  - Agregó manejo de error por cada expresión booleana.

12.3.5 Caso 5:

- Símbolos Obtenidos:

$$Salt = \{if(F^{\top})then(pf1_F : x/xs_F \wedge pfe_Fe : x/xs_Fe \wedge ra2_Apf1_F \wedge ra1_A :: y/yl_A \wedge l_A :: i_A/A \wedge ra2_A :: y/yl_A \wedge l_A :: i_A/A \wedge pae_Ae : x/xs_Ae)else(re1_Epfe_Fe \wedge re1_E :: y/yl_E \wedge l_E :: i_E/E)\} \wedge [if\exists X((X = pm1)then\{if(M^{\top})then(ra1_A :: y/yl_A \wedge l_A :: i_A/A \wedge ra2_A :: y/yl_A \wedge l_A :: i_A/A \wedge pae_Ae : x/xs_Ae \wedge pm1_M : x/xs_M \wedge pme_Me : x/xs_Me \wedge ra1_Apm1_M)else(re1_Epme_Me \wedge re1_E :: y/yl_E \wedge l_E :: i_E/E)\}) \ldots ((X = pn1)then\{if(N^{\top})then(ra1_A :: y/yl_A \wedge l_A :: i_A/A \wedge ra2_A :: y/yl_A \wedge l_A :: i_A/A \wedge pae_Ae : x/xs_Ae \wedge pn1_N : x/xS_N \wedge pne_Ne : x/xs_Ne \wedge ra1_Apn1_N)else(re1_Epne_Ne \wedge re1_E :: y/yl_E \wedge l_E :: i_E/E)\}) else(re1_Epae_Ae \wedge re1_E :: y/yl_E \wedge l_E :: i_E/E)]$$

- Conclusión: Este caso de prueba es una modificación del caso de prueba 4. Los cambios específicos fueron la agregación del componente que maneja el error, y un componente sencillo que está conectado al componente A. Teniendo en cuenta esto y comparando el resultado de este caso con el resultado del caso 4 se puede decir que:

    - El manejo de error se muestra correctamente.
    - Los componentes opcionales también están de manera correcta.
    - El componente sencillo tiene una estructura clara y está de acuerdo a la arquitectura de entrada.
    - Con lo anterior se concluye que el resultado es una especificación correcta de la arquitectura de entrada.

De lo anterior se observa que no hay exactitud total de los resultados obtenidos con respecto a los esperados. La justificación de esta falta de exactitud está dada teniendo en cuenta los antecedentes del desarrollo del prototipo.

Durante la investigación, se encontraron dos posibilidades para mostrar el resultado de las traducciones: en lenguaje HTML o a través de comandos de TeX. Aunque ambos soportan el total de símbolos propios de la

sintáxis del cálculo formal $-\rho_{arq}$, los APIs de Java que proveen las librerias necesarias para realizar la traducción correspondiente a cada una de estos lenguajes no soportan el total de dichos símbolos.

De los analizados al ejecutarse la traducción de un documento XML válido con respecto al esquema transformerumlarchro.xsd, el API proveído con el Proyecto JMathTeX es el que presenta la salida que más se ajusta a los resultados esperados. De ahí que se haya decidido éste lenguaje y no otro para la presentación de los resultados de las traducciones que se realicen.

Finalmente se concluye que a pesar de que no hay una concordancia del 100 % con los resultados mostrados en [Diosa, 2005], teniendo en cuenta las razones expuestas anteriormente, si hay concordancia fiel con los conceptos que contiene cada caso de prueba, el manejo de error, el manejo de componentes opcionales y sencillos y la combinación de estos conceptos.

13 CONCLUSIONES

- Se estableció como función principal la traducción desde xADL 2.0 al cálculo formal $-\rho_{arq}$ y se aclara que dentro del diseño del software no está contemplada la traducción inversa, es decir, que no se traducirá desde el cálculo formal $-\rho_{arq}$ a xADL 2.0.
- Se ha planteado que la edición de los documentos que maneja la aplicación (documento XML de entrada y documento con código LaTex de salida) no es una función primaria del software por lo tanto no se contempla en su diseño.
- Se estableció que el contenido del documento traducido será en código LaTeX, pues éste soporta toda la simbología utilizada en la sintaxis del cálculo formal – ρ_{arq}.
- El traductor a desarrollar permitirá generar tanto un documento de texto plano con contenido en código LaTeX, como una imagen en formato png con la interpretación de dicho código, es decir, con el resultado en simbología del cálculo formal – ρ_{arq}, para mayor comprensión de la sintaxis de dicho cálculo por parte del usuario.
- A pesar de los resultados de los casos de pruebas no fueron 100% fieles a los resultados esperados, los obtenidos reflejan que se soportan claramente los conceptos planteados por el cálculo Rho Arqu.

14 TRABAJOS FUTUROS

Con el resultado obtenido a partir de la propuesta planteada con éste proyecto, que se limita a la traducción sintáctica a partir de una descripción de una arquitectura de software desde un LDA hacia una especificación en el cálculo formal $-\rho_{arq}$, surgen algunos trabajos complementarios que aportarían a la investigación y al desarrollo de nuevas tecnologías a partir de metodologías formales y estructuradas para la industria del software.

El primero de estos trabajos consite en lograr que la concordancia entre los símbolos resultantes de la traducción hecha con el prototipo desarrollado, con respecto a los símbolos esperados con la traducción original, sea del 100%. Posterior a éste trabajo, una contribución significativa podría consistir en un desarrollo que permita realizar una traducción de la especificación de una arquitectura de software desde el cálculo formal $-\rho_{arq}$ hacia su descripción en un ADL, es decir, la traducción inversa de lo propuesto en éste proyecto en particular, permitiendo así verificar la estructura estática de una arquitectura de software a partir de la definición sintáctica de su comportamiento.

Otro aporte interesante surge de tomar como insumo el resultado obtenido con una traducción sintáctica como la propuesta en éste proyecto, para lograr la especificación de los aspectos dinámicos de la arquitectura a partir de la reducción semántica, permitiendo así controlar el comportamiento de sus componentes/conectores en tiempo de ejecución, haciendo posible ver el comportamiento de tal arquitectura de manera dinámica.

15 ANEXOS

15.1 MANUAL DEL USUARIO

A continuación se exponen los ítems necesarios para la instalación y correcto funcionamiento del software, además de una guía de cómo debe usarse para llevar a cabo el proceso de transformación de un archivo de entrada XML.

15.1.1 CREACIÓN DOCUMENTO XML PARA TRADUCCIÓN

Para la creación de un archivo XML de entrada, se debe tener en cuenta la estructura ya explicada del esquema transformerUmlarchro.xsd (Sección 7.3.1). Sin embargo, para una mejor comprensión, a continuación explica la correcta creación, a partir de un gráfico de un sistema sencillo.

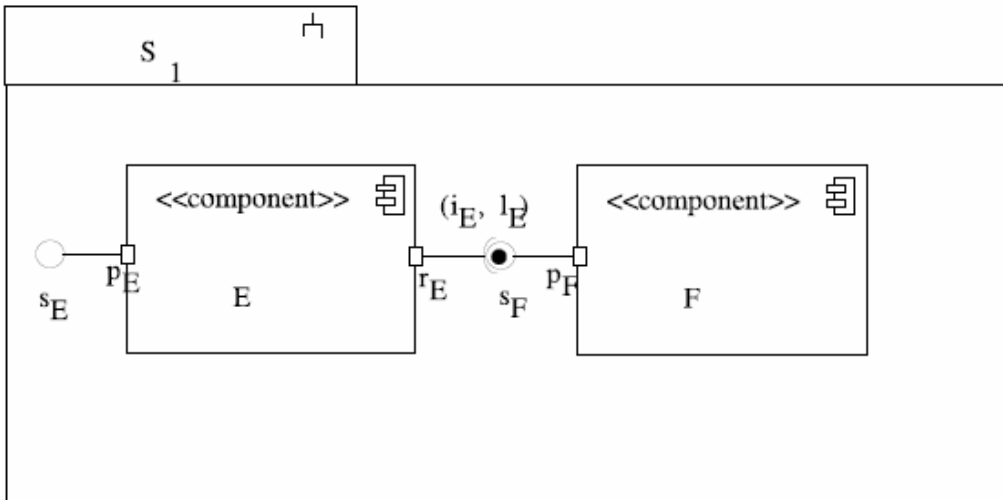

Figura 15.1 Gráfico Descripción Arquitectura Componente-Conector-Componente

La estructura del esquema transformerUmlarchro.xsd dice que el elemento raíz del XML a crear debe ser

transforUmlstructure:transforArchRo, el cual debe tener un atributo de tipo id, el cual debe ser único dentro del sistema a especificar. Así pues el principio del documento XML de entrada debería quedar de la siguiente manera:

```
<?xml version="1.0" encoding="UTF-8"?>
<archinstance:xArch xmlns:archinstance="instance.xsd"
xmlns:archtypes="types.xsd" xmlns:boolguard="boolguard.xsd"
xmlns:umlstructure="umlarchro.xsd"
xmlns:transforUmlstructure="transformerUmlarchro.xsd"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="transformerUmlarchro.xsd
transformerUmlarchro.xsd">
<transforUmlstructure:transforArchRo
transforUmlstructure:id="S1">
</ transforUmlstructure:transforArchRo>
```

El principio del encabezado esta dado por los esquemas que usa el esquema transformerUmlarchro.xsd y esto es necesario para la validación del documento de entrada. A continuación, dentro del elemento raíz, se debería colocar un elemento descripción que es obligatorio.

```
<transforUmlstructure:transforArchRo transforUmlstructure:id="S1">
<transforUmlstructure:description>S1</transforUmlstructure:descrip
tion>
</ transforUmlstructure:transforArchRo>
```

Luego de esto iniciamos con la especificación de los componentes que hay dentro de nuestro sistema. Así pues vendrían dos elementos de tipo transforUmlstructure:transformComponentRo para describir los componentes E y F respectivamente. Para un elemento de tipo transforUmlstructure:transformComponentRo la estructura es la siguiente:

- Atributo archtypes:id: Es obligatorio y único dentro del sistema.

- Elemento umlstructure:portRo: Existen por cada puerto que tenga el componente. Pueden ser cero o muchos elementos de este tipo. Internamente el port:Ro tiene la siguiente estructura:
    - Atributo archtypes:id: Es el id del puerto y es único dentro del sistema. Es obligatorio.
    - Elemento archtypes:description: Breve descripción del puerto. Es obligatorio.
    - Elemento umlstructure:parameter: El parámetro que se envía a través del puerto. Almacena el id del puerto.
    - Elemento umlstructure:subindex: Se refiere al subíndice del puerto. Almacena el mismo id del componente al que le pertenece el puerto.
    - Elemento umlstructure:umlType: Es el tipo de puerto, varía entre los valores provided o required dependiendo si el puerto es de provisión o requerimiento.
    - Elemento umlstructure:interfaceRo: Todo puerto tiene una interfaz relacionada. Este elemento representa esa interfaz y contiene un atributo de tipo archtypes:id que es único y obligatorio. También tiene un elemento de tipo archtypes:description que contiene una breve descripción de la interfaz. Contiene un elemento de tipo umlstructure:umlType el cual toma valores provided y required dependiendo si el puerto al que pertenece es de provisión o de requerimiento. La interfaceRo también contiene un elemento umlstructure:parameter que contiene el valor de lo que se envía ó recibe a través de esa interfaz. Finalmente contiene un elemento de tipo umlstructure:subindex que contiene el id del componente al cual pertenece. Todos los elementos de la interfaceRo son obligatorios.

- Elemento transforUmlstructure:rolRo: Es el rol del componente, y es obligatorio. Puede tomar los valores de:
    - initial: Componente que inicia el flujo de datos en el sistema.
    - intermediate: Componente que participa en el flujo de datos del sistema.
    - errorHandler: Componente que se encarga del manejo de errores dentro del sistema.
    - final: Componente que finaliza el flujo de datos del sistema.
    - option: Componente opcional dentro de un sistema.
- Elemento transforUmlstructure:expBoolRo: Es un elemento que se encarga de manejar el concepto de manejo de error dentro del sistema para cada componente. No es obligatorio. Este elemento tiene internamente dos elementos de tipo transforUmlstructure:referenceRo. El referenceRo tiene la siguiente estructura interna:
    - Atributo transforUmlstructure:referenciaRo: Es un atributo que contiene el id de un componente dentro del sistema.
    - Elemento transforUmlstructure:estado: Es un elemento que varía entre los valores de “false” y “true”. Así pues si un sistema tiene manejo de error entonces debe tener un transforUmlstructure:expBoolRo donde el primer elemento interno transforUmlstructure:referenceRo referencia al componente de éxito y el segundo elemento referencia al componente de manejo de error.
- Elemento transforUmlstructure:variableSet: Se encarga del manejo de variables para componentes opcionales o alternativos. No es obligatorio e internamente contiene un elemento de tipo transforUmlstructure:variable que contiene el nombre de la variable.

Teniendo en cuenta la anterior descripción de un componente a través de la especificación XML, los componentes E y F quedarían de la siguiente forma:

Componente E:

```
<transforUmlstructure:transformComponentRo archtypes:id="E">
    <archtypes:description>E</archtypes:description>
    <umlstructure:portRo archtypes:id="pe1">
        <archtypes:description>puerto provision</archtypes:description>
        <umlstructure:interfaceRo archtypes:id="pe1.1">
            <archtypes:description>Interface de provision</archtypes:
            description>
            <umlstructure:umlType>provided</umlstructure:umlType>
            <umlstructure:parameter>S</umlstructure:parameter>
            <umlstructure:subindex>E</umlstructure:subindex>
        </umlstructure:interfaceRo>
        <umlstructure:parameter>pe1</umlstructure:parameter>
        <umlstructure:subindex>E</umlstructure:subindex>
        <umlstructure:umlType>provided</umlstructure:umlType>
    </umlstructure:portRo>
    <umlstructure:portRo archtypes:id="re1">
        <archtypes:description>Puerto de requerimiento</archtypes:
        description>
        <umlstructure:interfaceRo archtypes:id="re1.1">
            <archtypes:description>Interfaz de requerimiento</archtypes:
            description>
            <umlstructure:umlType>required</umlstructure:umlType>
            <umlstructure:parameter>i</umlstructure:parameter>
            <umlstructure:parameter>l</umlstructure:parameter>
            <umlstructure:subindex>E</umlstructure:subindex>
        </umlstructure:interfaceRo>
        <umlstructure:parameter>re1</umlstructure:parameter>
        <umlstructure:subindex>E</umlstructure:subindex>
        <umlstructure:umlType>required</umlstructure:umlType>
    </umlstructure:portRo>
<transforUmlstructure:rolRo>intermediate</transforUmlstructure:rolRo>
    </transforUmlstructure:transformComponentRo>
```

Componente F:

```
<transforUmlstructure:transformComponentRo archtypes:id="F">
```

```
            <archtypes:description>F</archtypes:description>
            <umlstructure:portRo archtypes:id="pf1">
                <archtypes:description>Puerto de provision 1</archtypes:
                description>
                <umlstructure:interfaceRo archtypes:id="pf1.1">
                    <archtypes:description>Interfaz de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>F</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pf1</umlstructure:parameter>
                <umlstructure:subindex>F</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <transforUmlstructure:rolRo>initial</transforUmlstructure:rolRo>
        </transforUmlstructure:transformComponentRo>
        <transforUmlstructure:connectorRo archinstance:id="C1">
            <archinstance:description>Conector entre E y F</archinstance:
            description>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="re1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pf1" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
    </transforUmlstructure:transforArchRo>
</archinstance:xArch>
```

Para completar la especificación en XML de la arquitectura de entrada mostrada en la figura tal, es necesario ahora hacer el código correspondiente a la especificación del conector entre los componentes E y F. El conector está representado por el elemento transforUmlstructure:connectorRo que contiene la siguiente estructura interna:

- Atributo archtypes:id: Es único dentro del sistema y es obligatorio para cualquier conector.
- Elemento archtypes:description: Es obligatorio y contiene una breve descripción del conector.
- Elemento archinstance:point: Un conector debe tener dos elementos de este tipo, y su función es guardar la referencia a los dos componentes que esta conectando. Este elemento internamente contiene:
    - Elemento archinstance:anchorOnInterface: Este elemento tiene dos atributos, el xlink:href y el xlink:type. En el primero, se guarda el id del puerto de uno de los componentes a conectar, y en el segundo atributo se guarda el tipo de link que es, que para nuestro caso tiene el valor de “simple”. Para nuestro caso, el primer archinstance:point en su elemento interno archinstance:anchorOnInterface debe guardar el id del puerto de requerimiento del componente E. El segundo elemento archinstance:point en su elemento interno archinstance:anchorOnInterface debe guardar el id del puerto de provisión de F. Siempre el orden de creación de los puntos debe ser desde el puerto de requerimiento a el puerto de provisión.

Así pues, la especificación del conector entre E y F quedara de la siguiente forma:

Conector E y F:

```
<transforUmlstructure:connectorRo archinstance:id="C1">
    <archinstance:description>Conector entre E y F</archinstance:
    description>
    <archinstance:point>
```

```
                <archinstance:anchorOnInterface xlink:href="re1" xlink:type=
                "simple"/>
            </archinstance:point>
            <archinstance:point>
                <archinstance:anchorOnInterface xlink:href="pf1" xlink:type=
                "simple"/>
            </archinstance:point>
        </transforUmlstructure:connectorRo>
```

Teniendo en cuenta los resultados anteriores, la especificación completa sería la siguiente:

```
<archinstance:xArch xmlns:archinstance="instance.xsd"
xmlns:archtypes="types.xsd" xmlns:boolguard="boolguard.xsd"
xmlns:umlstructure="umlarchro.xsd"
xmlns:transforUmlstructure="transformerUmlarchro.xsd"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="transformerUmlarchro.xsd transformerUmlarchro.xsd">
    <transforUmlstructure:transforArchRo transforUmlstructure:id="S2">
        <transforUmlstructure:description>S2</transforUmlstructure:description>
        <transforUmlstructure:transformComponentRo archtypes:id="E">
            <archtypes:description>E</archtypes:description>
            <umlstructure:portRo archtypes:id="pe1">
                <archtypes:description>puerto provision</archtypes:description>
                <umlstructure:interfaceRo archtypes:id="pe1.1">
                    <archtypes:description>Interface de provision</archtypes:
                    description>
                    <umlstructure:umlType>provided</umlstructure:umlType>
                    <umlstructure:parameter>S</umlstructure:parameter>
                    <umlstructure:subindex>E</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>pe1</umlstructure:parameter>
                <umlstructure:subindex>E</umlstructure:subindex>
                <umlstructure:umlType>provided</umlstructure:umlType>
            </umlstructure:portRo>
            <umlstructure:portRo archtypes:id="re1">
                <archtypes:description>Puerto de requerimiento</archtypes:
                description>
```

```
                <umlstructure:interfaceRo archtypes:id="re1.1">
                    <archtypes:description>Interfaz de requerimiento</archtypes:
                    description>
                    <umlstructure:umlType>required</umlstructure:umlType>
                    <umlstructure:parameter>i</umlstructure:parameter>
                    <umlstructure:parameter>l</umlstructure:parameter>
                    <umlstructure:subindex>E</umlstructure:subindex>
                </umlstructure:interfaceRo>
                <umlstructure:parameter>re1</umlstructure:parameter>
                <umlstructure:subindex>E</umlstructure:subindex>
                <umlstructure:umlType>required</umlstructure:umlType>
            </umlstructure:portRo>
        <transforUmlstructure:rolRo>intermediate</transforUmlstructure:rolRo>
    </transforUmlstructure:transformComponentRo>
    <transforUmlstructure:transformComponentRo archtypes:id="F">
        <archtypes:description>F</archtypes:description>
        <umlstructure:portRo archtypes:id="pf1">
            <archtypes:description>Puerto de provision 1</archtypes:
            description>
            <umlstructure:interfaceRo archtypes:id="pf1.1">
                <archtypes:description>Interfaz de provision</archtypes:
                description>
                <umlstructure:umlType>provided</umlstructure:umlType>
                <umlstructure:parameter>S</umlstructure:parameter>
                <umlstructure:subindex>F</umlstructure:subindex>
            </umlstructure:interfaceRo>
            <umlstructure:parameter>pf1</umlstructure:parameter>
            <umlstructure:subindex>F</umlstructure:subindex>
            <umlstructure:umlType>provided</umlstructure:umlType>
        </umlstructure:portRo>
        <transforUmlstructure:rolRo>initial</transforUmlstructure:rolRo>
    </transforUmlstructure:transformComponentRo>
    <transforUmlstructure:connectorRo archinstance:id="C1">
        <archinstance:description>Conector entre E y F</archinstance:
        description>
        <archinstance:point>
            <archinstance:anchorOnInterface xlink:href="re1" xlink:type=
            "simple"/>
        </archinstance:point>
        <archinstance:point>
            <archinstance:anchorOnInterface xlink:href="pf1" xlink:type=
            "simple"/>
        </archinstance:point>
    </transforUmlstructure:connectorRo>
```

```
    </transforUmlstructure:transforArchRo>
</archinstance:xArch>
```

15.1.2 INSTALACIÓN

15.1.2.1 Requerimientos Técnicos

Para que la aplicación funcione, la maquina en donde se hace la instalación debe tener un hardware similar o superior al siguiente:

- Espacio libre en disco de 20 MB.
- Un mínimo de memoria RAM de 128 MB.

La aplicación es un desarrollo J2SE, y para poder ejecutarlo, ese necesario tener pre-instalada la maquina virtual de java. Este software puede descargarse de: www.java.com/es/download.

15.1.2.2 xADL2.0toRhoArq

Como tal, la aplicación viene empaquetada en un instalador automático, el cual colocara accesos directos al archivo .jar que es similar a un ejecutable .exe.

La instalación de la aplicación es similar a una aplicación estándar para Windows. Sin embargo la aplicación puede ser ejecutada en otros sistemas operativos:

Figura 15.2 Pantalla de Instalación de la Aplicación

En un sistema operativo Windows, la aplicación quedaría ubicada en los archivos de programa, y en el menú inicio como se muestra a continuación:

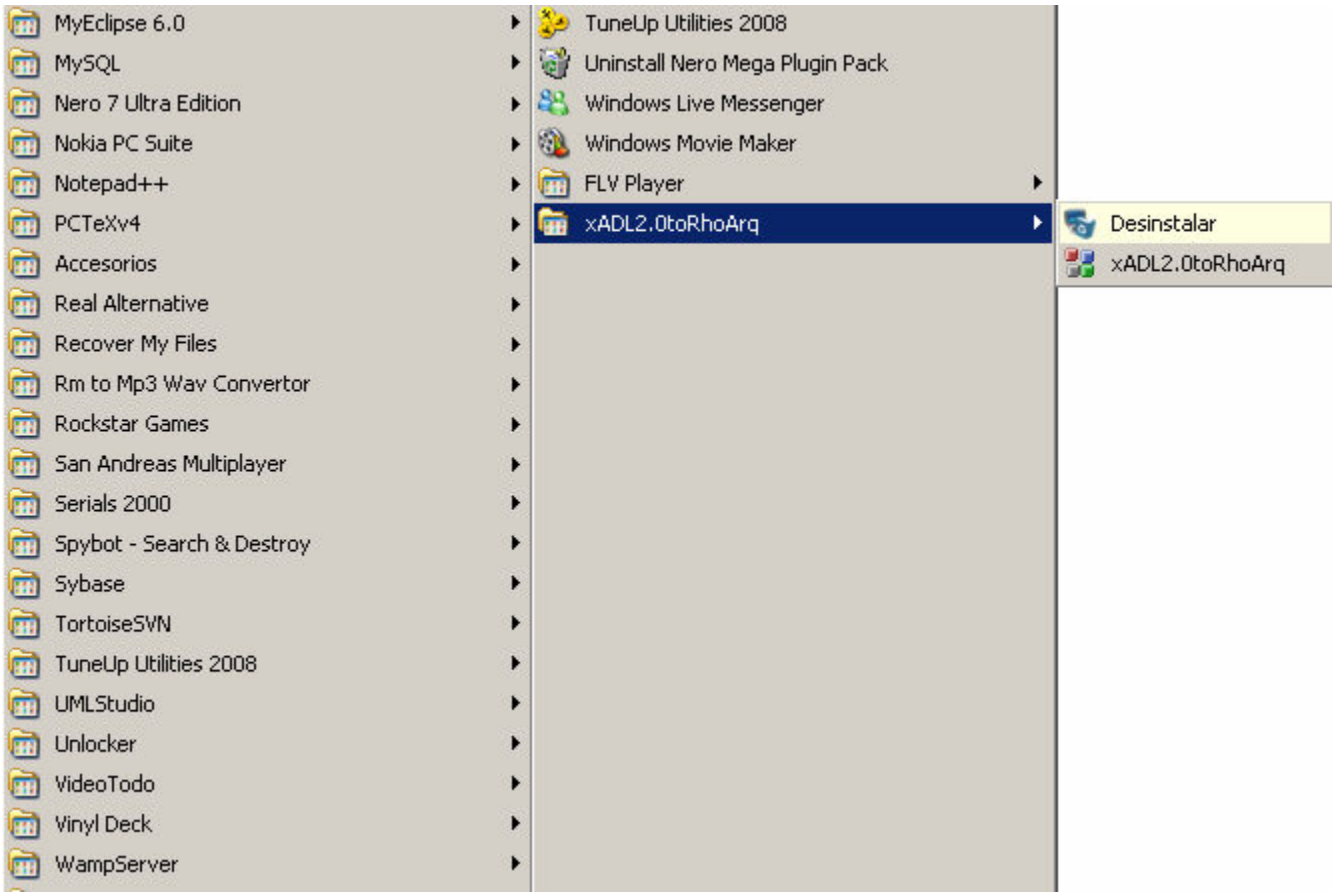

Figura 15.3 Pantalla Ubicación Aplicación Sistema Windows

Como se puede observar, además de mostrarse el ejecutable para la aplicación se da la opción de desinstalar la misma del equipo.

En este momento podremos empezar a utilizar la aplicación, ejecutando procesos de Importación, validación y traducción de archivos XML.

Por último, para verificar los datos de creación del proyecto, se puede consultar la opción Acerca de… del menú Ayuda, para lo cual la aplicación mostrará lo siguiente.

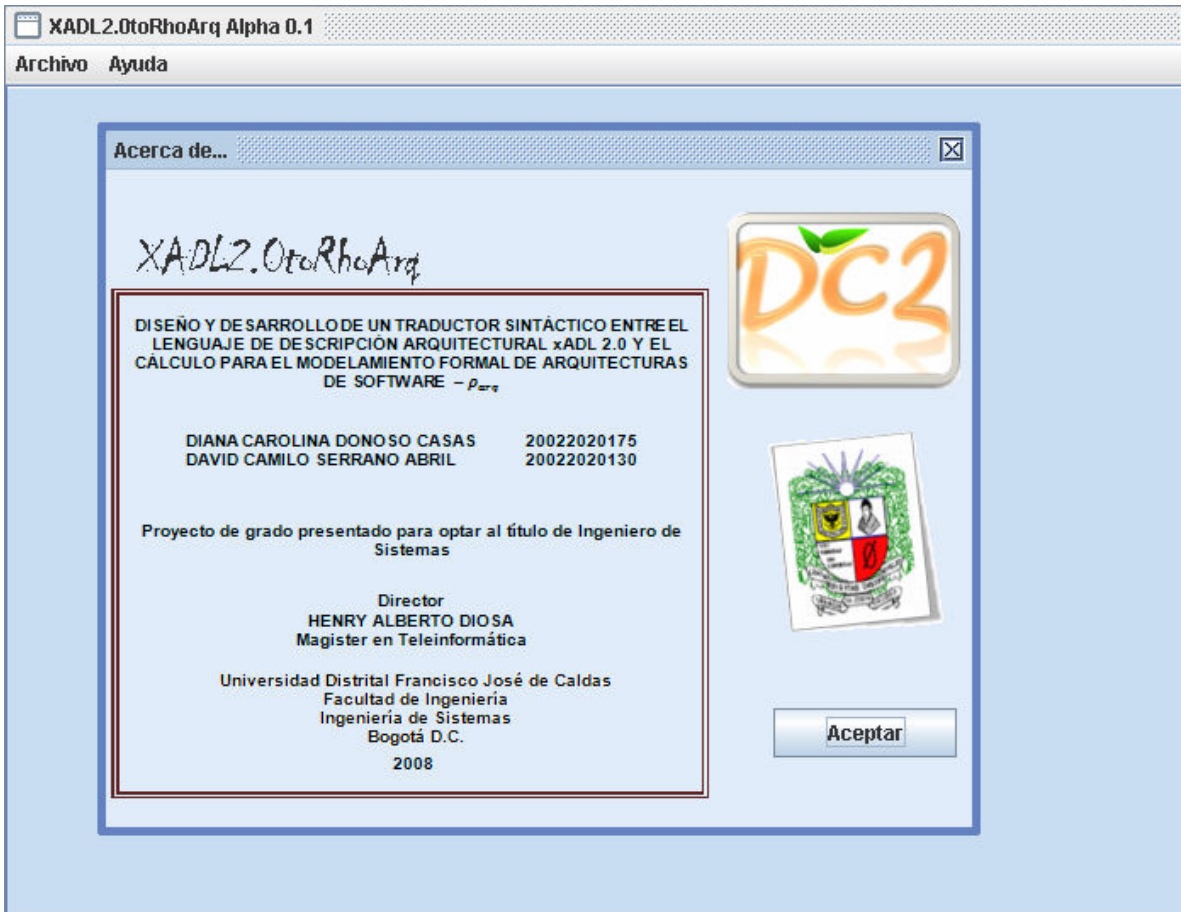

Figura 15.4 Pantalla Acerca de...

15.1.3 FUNCIONAMIENTO

Una vez instalada la aplicación en el equipo donde se desea utilizar, teniendo en cuenta que ésta permite realizar la traducción de un documento XML que contenga la especificación de una arquitectura en lenguaje xADL2.0 extendido a una especificación formal basada en reglas sintácticas del cálculo – ρ_{arq}, es necesario contar con dicho documento de entrada. Para la elaboración del documento XML de entrada, consulte la sección 15.1.1.

15.1.3.1 Importar documento XML de entrada

La función **Importar** permite seleccionar un documento XML válido con respecto al esquema de xADL 2.0 extendido para ser traducido.

Para importar un archivo, seleccione la opción **Importar** del menú **Archivo** en la barra de menús, tal y como se muestra a continuación.

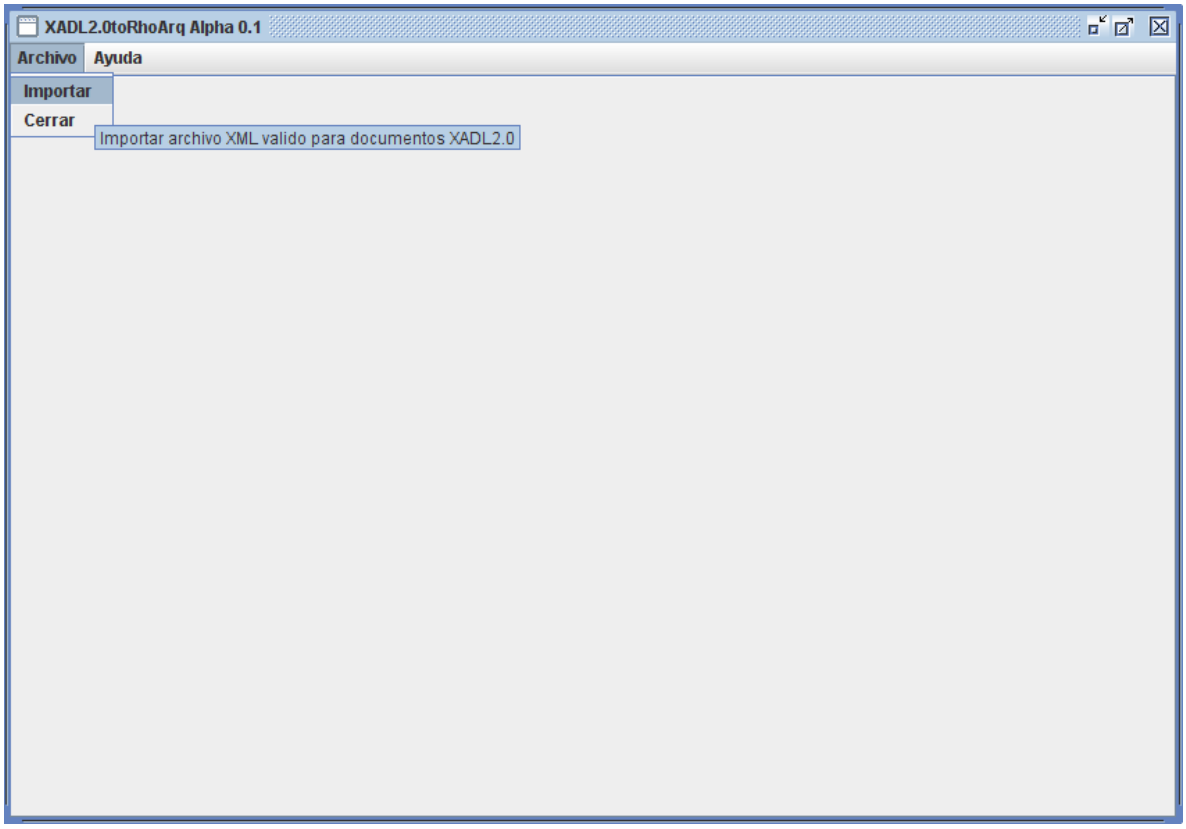

Figura 15.5 Pantalla Menú Archivo opción Importar

A continuación se mostrará una ventana emergente en donde se debe seleccionar la ruta en donde se encuentra el documento que se desea traducir.

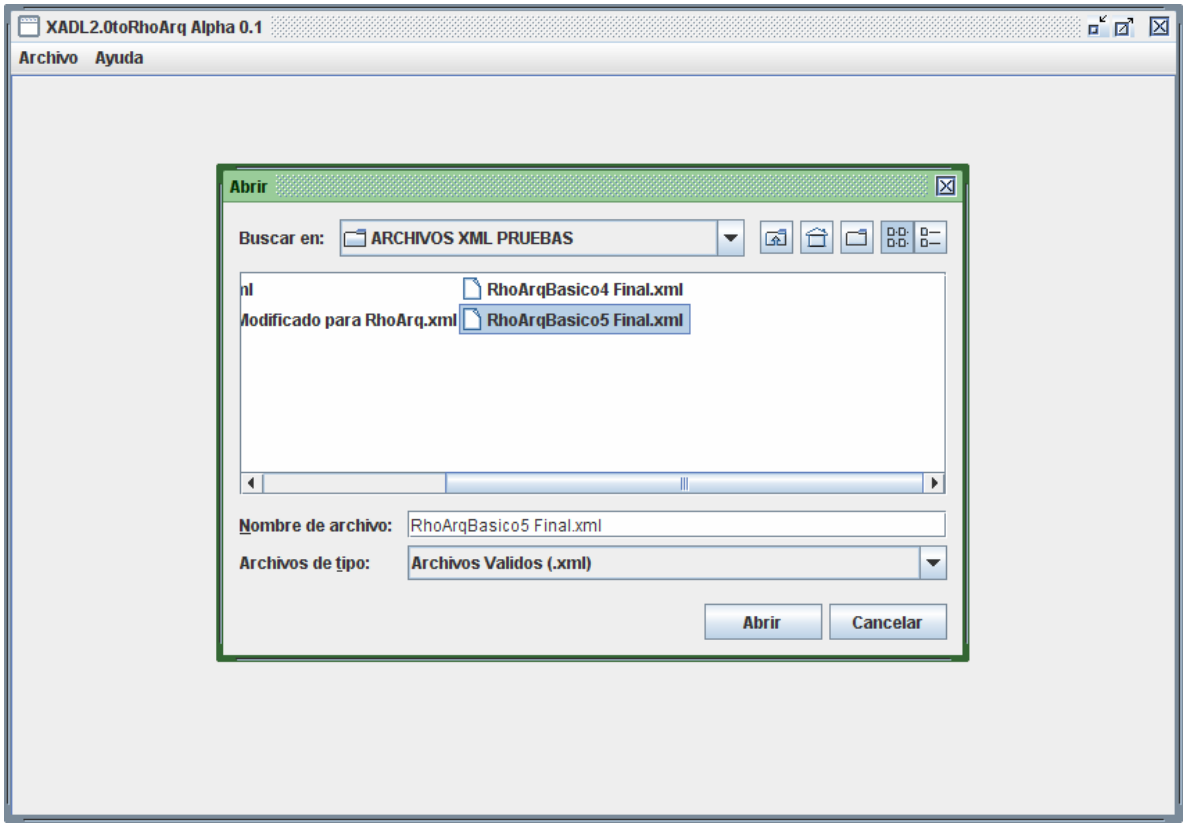

Figura 15.6 Pantalla Selección Documento XML a Traducir

La aplicación únicamente permite importar documentos con extensión *.xml, siempre y cuando éstos sean válidos con respecto al esquema xADL2.0 extendido (transformerUmlarchro.xsd Sección 7.3.1). Si el archivo seleccionado no es un documento XML o no es válido, la aplicación muestra un mensaje indicando que no se puede realizar la importación.

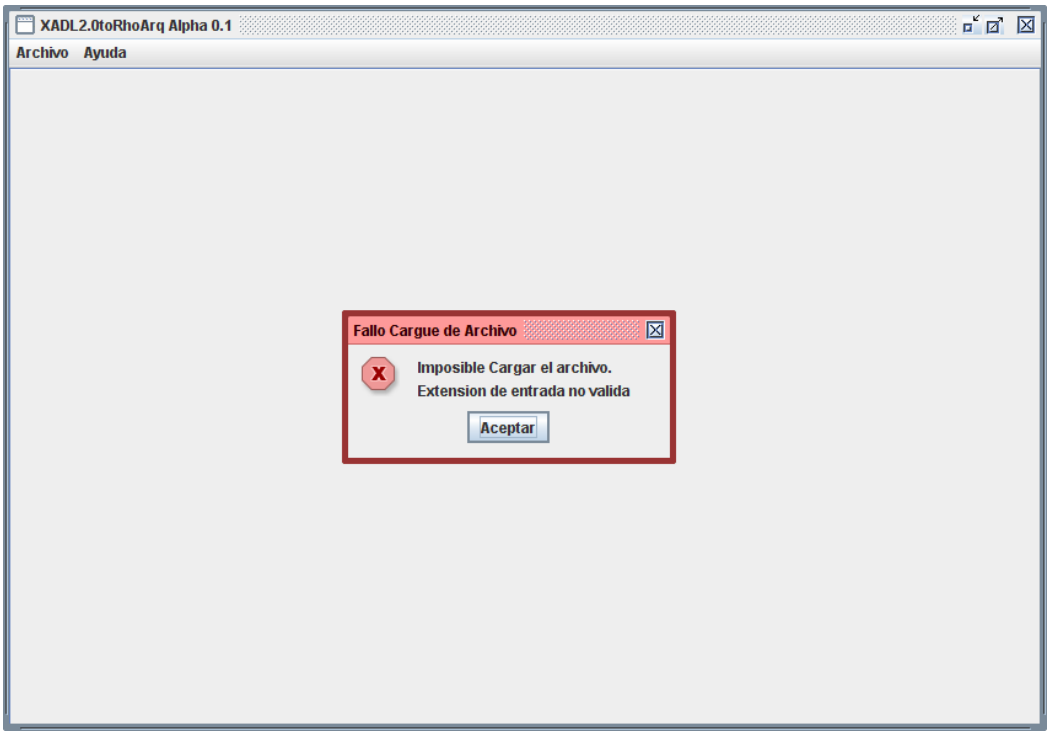

Figura 15.7 Pantalla Extensión Inválida del Archivo a Importar

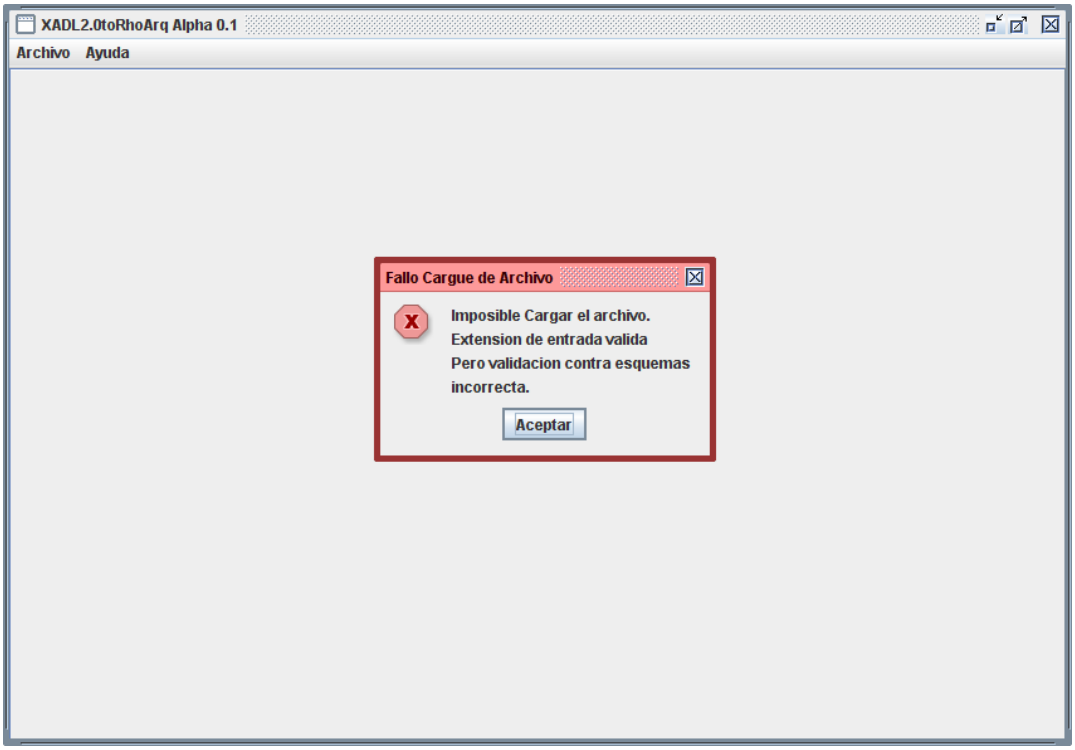

Figura 15.8 Pantalla Documento Inválido Contra Esquemas

Si la importación del documento XML fue exitosa, es decir, la extensión del documento es *.xml y además es válido con respecto al esquema transformerUmlarchro.xsd (Sección 7.3.1), la aplicación muestra un mensaje de éxito en el cargue del archivo

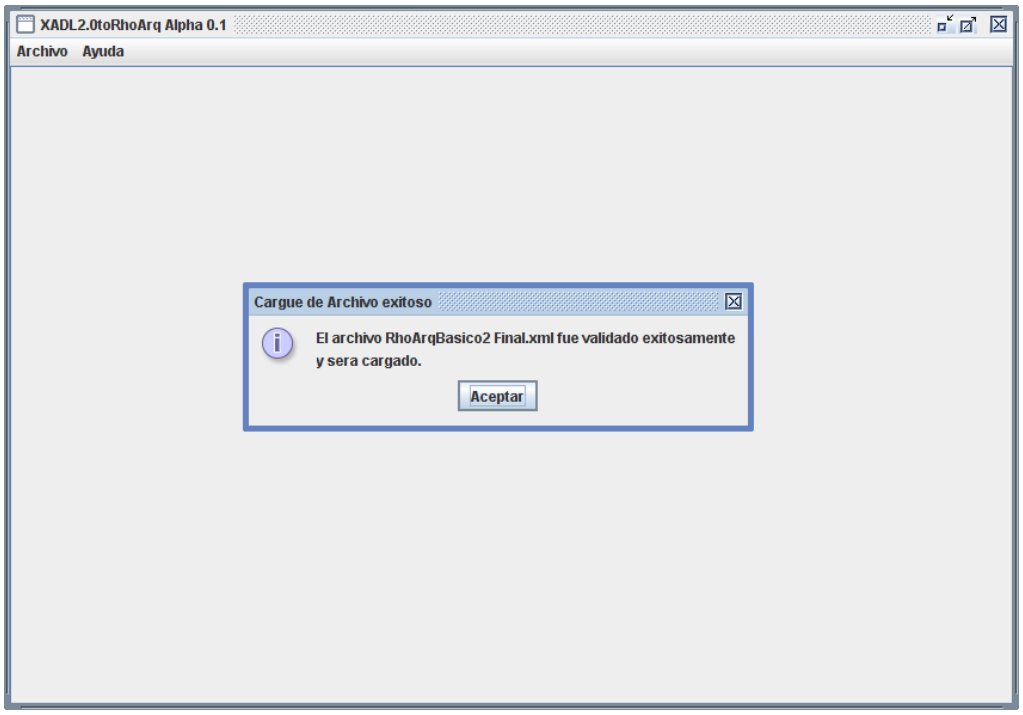

Figura 15.9 Pantalla Exito al Importar Archivo

y a continuación muestra el documento importado en un tab del área de trabajo con las opciones de **Traducir** y **Cerrar** el archivo importado, como se muestra a continuación.

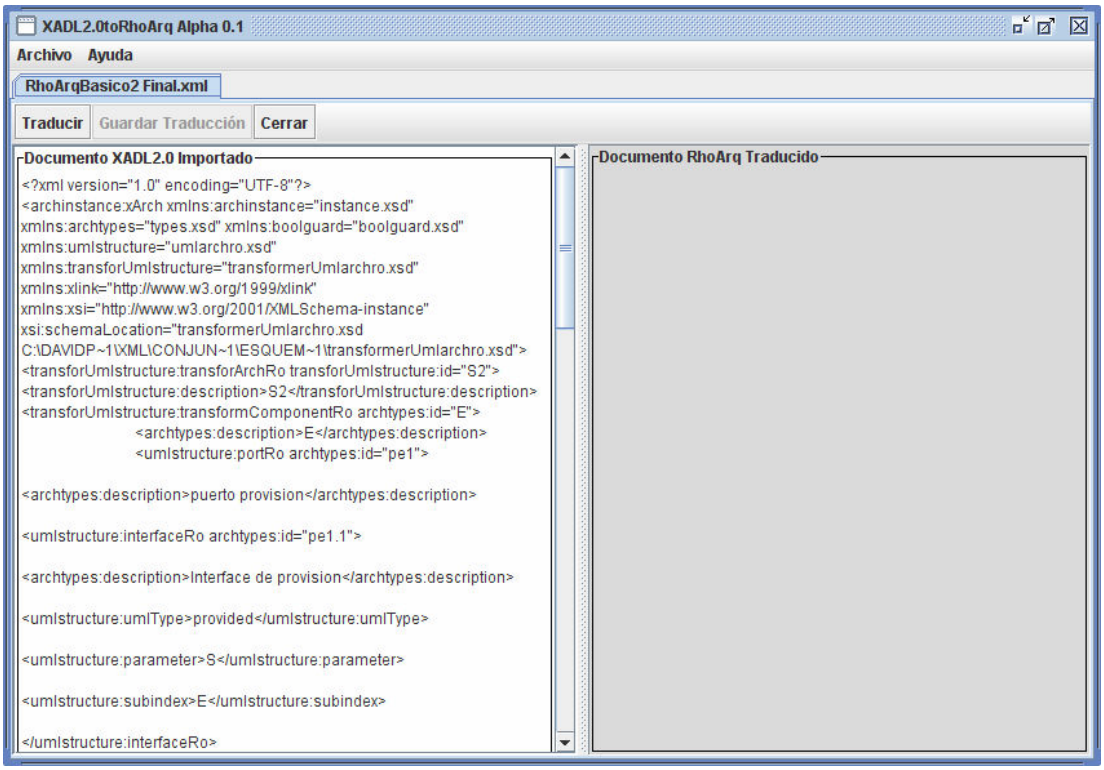

Figura 15.10 Pantalla Documento XML Importado

La aplicación permite importar varios documentos XML válidos simultáneamente, los cuales carga en tabs separados:

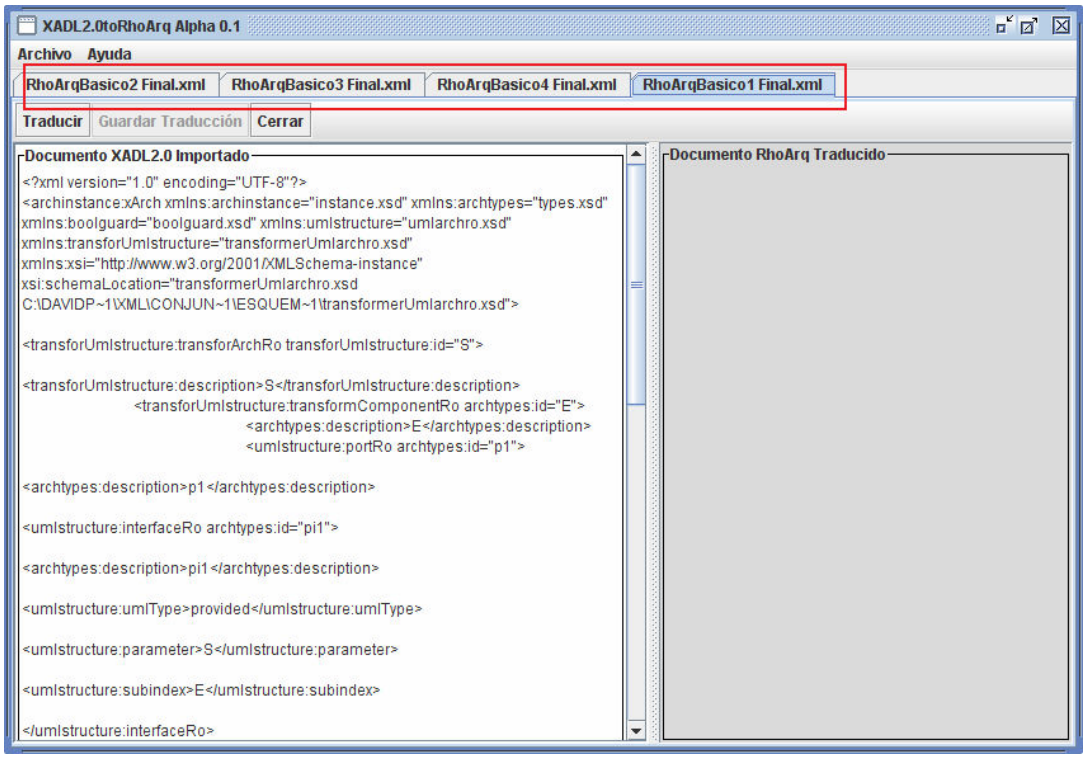

Figura 15.11 Pantalla Importar Varios Documentos XML Simultáneamente

15.1.3.2 Traducir documento XML importado a sintáxis RhoArq

Posterior a la importación del documento XML válido con respecto al esquema correspondiente, para realizar la traducción a código con sintáxis del cálculo RhoArq, se debe seleccionar la opción **Traducir** en el tab donde se encuentre cargado el documento XML correspondiente.

Una vez finalizada la traducción del documento seleccionado, se mostrará un mensaje de éxito de la operación

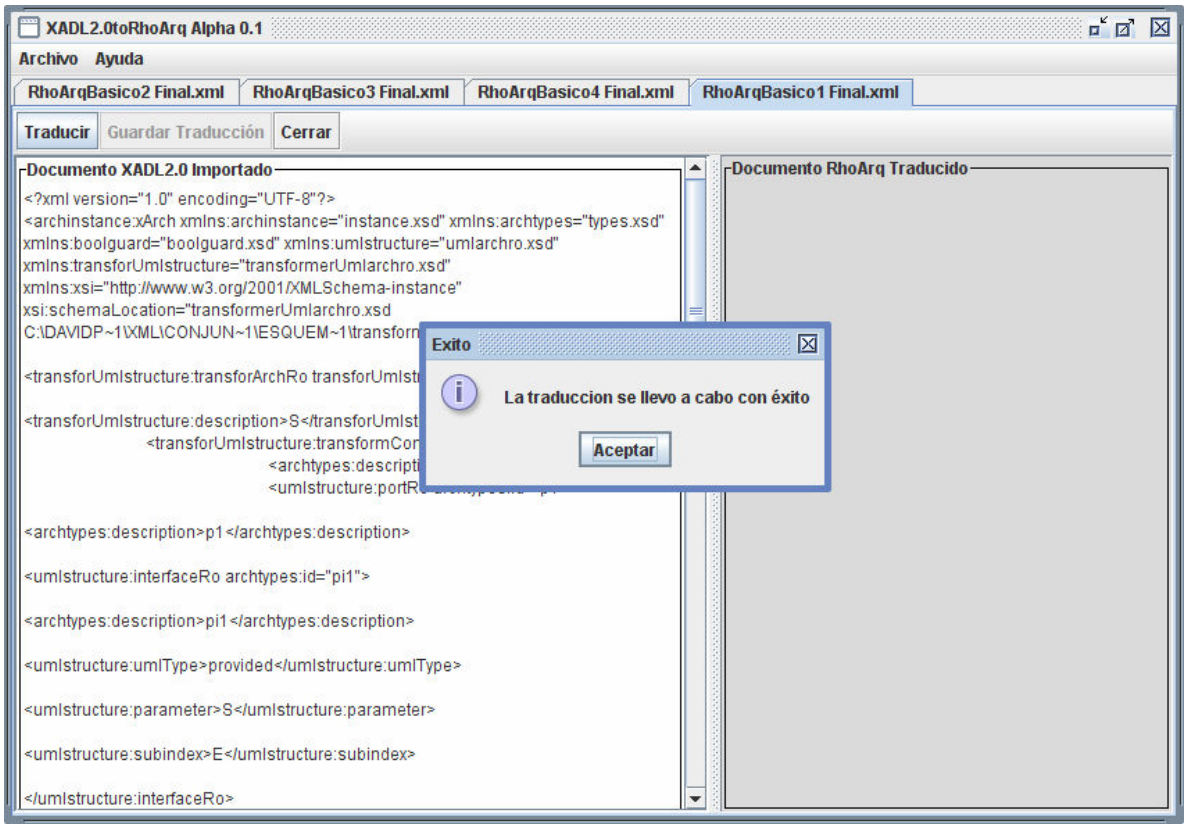

Figura 15.12 Pantalla Traducción Documento XML

y en seguida se muestra en el panel derecho del tab correspondiente al archivo, el resultado de la traducción.

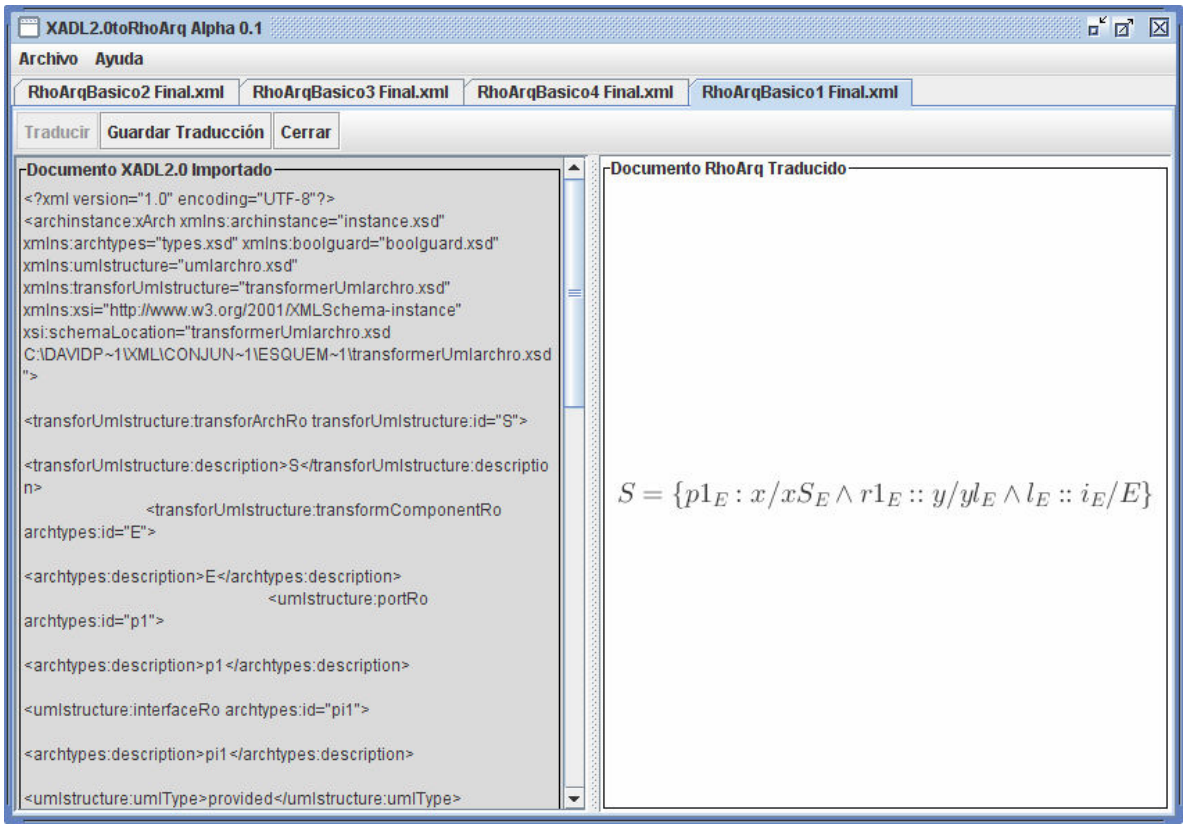

Figura 15.13 Pantalla Documento Traducido

Como se puede observar, después de llevar a cabo la traducción del documento XML, éste queda inactivo en el área de trabajo del tab correspondiente y se muestra activo el resultado de la traducción.

15.1.3.3 Guardar documento XML traducido

La opción de guardar archivo únicamente estará habilitado una vez se haya realizado la traducción de algún documento XML; ésta opción almacena una imagen en formato *.png y un documento en código LaTeX *.tex con el resultado de la traducción, en la ubicación seleccionada por el usuario.

Para llevar a cabo esta operación, debe seleccionar la opción **Guardar Traducción** de la barra de menús habilitada en el tab donde se encuentra abierto el documento que ha sido traducido.

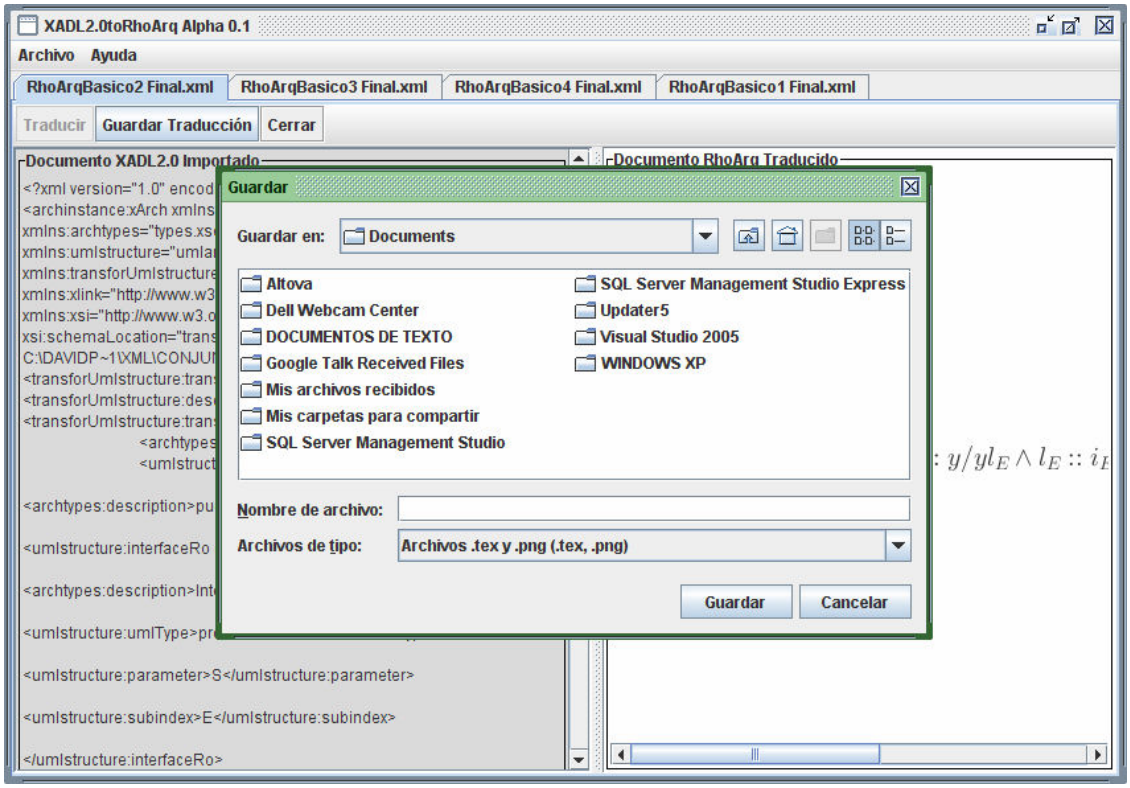

Figura 15.14 Pantalla Ubicación Guardar Documento Traducido

La aplicación muestra una ventana emergente para que el usuario seleccione la ubicación de los archivos que serán almacenados con el resultado de la traducción.

El usuario digita el nombre que desee asignar a los documentos correspondientes de la traducción realizada y la aplicación almacenará ambos documentos con el mismo nombre, pero con diferentes extensiones: *.png y *.tex.

15.2 JAVA DOCS

A continuación se detallan cada uno de los JavaDocs que fueron utilizados para el desarrollo de la aplicación, producto de éste proyecto:

15.2.1 Class File

java.lang.Object [SUN, 2, 2008]

└ **java.io.File**

All Implemented Interfaces:

Comparable, Serializable

---

public class **File**
extends Object
implements Serializable, Comparable

An abstract representation of file and directory pathnames.
User interfaces and operating systems use system-dependent *pathname strings* to name files and directories. This class presents an abstract, system-independent view of hierarchical pathnames. An *abstract pathname* has two components:

1. An optional system-dependent *prefix* string, such as a disk-drive specifier, "/" for the UNIX root directory, or "\\" for a Microsoft Windows UNC pathname, and
2. A sequence of zero or more string *names*. Each name in an abstract pathname except for the last denotes a directory; the last name may denote either a directory or a file. The *empty* abstract pathname has no prefix and an empty name sequence.

The conversion of a pathname string to or from an abstract pathname is inherently system-dependent. When an abstract pathname is converted into a pathname string, each name is separated from the next by a single copy of the default *separator character*. The default name-separator character is defined by the system property file.separator, and is made available in the public static fields separator and separatorChar of this class. When a pathname string is converted into an abstract pathname, the names within it may be separated by the default name-separator character or by any other name-separator character that is supported by the underlying system.

A pathname, whether abstract or in string form, may be either *absolute* or *relative*. An absolute pathname is complete in that no other information is required in order to locate the file that it denotes. A relative pathname, in contrast, must be interpreted in terms of information taken from some other pathname. By default the classes in the java.io package always resolve relative pathnames against the current user directory. This directory is named by the system property user.dir, and is typically the directory in which the Java virtual machine was invoked.

The prefix concept is used to handle root directories on UNIX platforms, and drive specifiers, root directories and UNC pathnames on Microsoft Windows platforms, as follows:

- For UNIX platforms, the prefix of an absolute pathname is always "/". Relative pathnames have no prefix. The abstract pathname denoting the root directory has the prefix "/" and an empty name sequence.

- For Microsoft Windows platforms, the prefix of a pathname that contains a drive specifier consists of the drive letter followed by ":" and possibly followed by "\" if the pathname is absolute. The prefix of a UNC pathname is "\\"; the hostname and the share name are the first two names in the name sequence. A relative pathname that does not specify a drive has no prefix.

  Instances of the File class are immutable; that is, once created, the abstract pathname represented by a File object will never change.

Since:

JDK1.0

See Also:

Serialized Form

Field Summary	
static String	pathSeparator The system-dependent path-separator character, represented as a string for convenience.
static char	pathSeparatorChar The system-dependent path-separator character.
static String	separator The system-dependent default name-separator character, represented as a string for convenience.
static char	separatorChar The system-dependent default name-separator character.

Constructor Summary
File(File parent, String child) Creates a new File instance from a parent abstract pathname and a child pathname string.
File(String pathname) Creates a new File instance by converting the given pathname string into an abstract pathname.

Constructor Summary
File(String parent, String child) Creates a new File instance from a parent pathname string and a child pathname string.
File(URI uri) Creates a new File instance by converting the given file: URI into an abstract pathname.

Method Summary	
boolean	canRead() Tests whether the application can read the file denoted by this abstract pathname.
boolean	canWrite() Tests whether the application can modify to the file denoted by this abstract pathname.
int	compareTo(File pathname) Compares two abstract pathnames lexicographically.
int	compareTo(Object o) Compares this abstract pathname to another object.
boolean	createNewFile() Atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist.
static File	createTempFile(String prefix, String suffix) Creates an empty file in the default temporary-file directory, using the given prefix and suffix to generate its name.
static File	createTempFile(String prefix, String suffix, File directory) Creates a new empty file in the specified directory, using the given prefix and suffix strings to generate its name.
boolean	delete() Deletes the file or directory denoted by this abstract pathname.
void	deleteOnExit() Requests that the file or directory denoted by this abstract pathname be deleted when the virtual machine terminates.
boolean	equals(Object obj) Tests this abstract pathname for equality with the given object.

Method Summary	
boolean	exists() Tests whether the file or directory denoted by this abstract pathname exists.
File	getAbsoluteFile() Returns the absolute form of this abstract pathname.
String	getAbsolutePath() Returns the absolute pathname string of this abstract pathname.
File	getCanonicalFile() Returns the canonical form of this abstract pathname.
String	getCanonicalPath() Returns the canonical pathname string of this abstract pathname.
String	getName() Returns the name of the file or directory denoted by this abstract pathname.
String	getParent() Returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory.
File	getParentFile() Returns the abstract pathname of this abstract pathname's parent, or null if this pathname does not name a parent directory.
String	getPath() Converts this abstract pathname into a pathname string.
int	hashCode() Computes a hash code for this abstract pathname.
boolean	isAbsolute() Tests whether this abstract pathname is absolute.
boolean	isDirectory() Tests whether the file denoted by this abstract pathname is a directory.
boolean	isFile() Tests whether the file denoted by this abstract pathname is a normal file.
boolean	isHidden() Tests whether the file named by this abstract pathname is a hidden file.

Method Summary	
long	lastModified() Returns the time that the file denoted by this abstract pathname was last modified.
long	length() Returns the length of the file denoted by this abstract pathname.
String[]	list() Returns an array of strings naming the files and directories in the directory denoted by this abstract pathname.
String[]	list(FilenameFilter filter) Returns an array of strings naming the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
File[]	listFiles() Returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname.
File[]	listFiles(FileFilter filter) Returns an array of abstract pathnames denoting the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
File[]	listFiles(FilenameFilter filter) Returns an array of abstract pathnames denoting the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
static File[]	listRoots() List the available filesystem roots.
boolean	mkdir() Creates the directory named by this abstract pathname.
boolean	mkdirs() Creates the directory named by this abstract pathname, including any necessary but nonexistent parent directories.
boolean	renameTo(File dest) Renames the file denoted by this abstract pathname.
boolean	setLastModified(long time) Sets the last-modified time of the file or directory named by this abstract pathname.

Method Summary	
boolean	setReadOnly() Marks the file or directory named by this abstract pathname so that only read operations are allowed.
String	toString() Returns the pathname string of this abstract pathname.
URI	toURI() Constructs a file: URI that represents this abstract pathname.
URL	toURL() Converts this abstract pathname into a file: URL.

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

15.2.2 Class Schema

java.lang.Object [SUN, 3, 2008]

└ **javax.xml.validation.Schema**

---

public abstract class **Schema**

extends Object

Immutable in-memory representation of grammar.

This object represents a set of constraints that can be checked/ enforced against an XML document.

A Schema object is thread safe and applications are encouraged to share it across many parsers in many threads.

A Schema object is immutable in the sense that it shouldn't change the set of constraints once it is created. In other words, if an application validates

the same document twice against the same Schema, it must always produce the same result.

A Schema object is usually created from SchemaFactory.

Two kinds of validators can be created from a Schema object. One is Validator, which provides highly-level validation operations that cover typical use cases. The other is ValidatorHandler, which works on top of SAX for better modularity.

This specification does not refine the Object.equals(java.lang.Object) method. In other words, if you parse the same schema twice, you may still get !schemaA.equals(schemaB).

Since:

1.5

See Also:

XML Schema Part 1: Structures, Extensible Markup Language (XML) 1.1, Extensible Markup Language (XML) 1.0 (Second Edition)

Constructor Summary	
protected	Schema() Constructor for the derived class.

Method Summary	
abstract Validator	newValidator() Creates a new Validator for this Schema.
abstract ValidatorHandler	newValidatorHandler() Creates a new ValidatorHandler for this Schema.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Schema

protected **Schema**()

Constructor for the derived class.

The constructor does nothing.

Method Detail

newValidator

public abstract Validator **newValidator**()

Creates a new Validator for this Schema.

A validator enforces/checks the set of constraints this object represents.

Returns:
Always return a non-null valid object.

---

newValidatorHandler

public abstract ValidatorHandler **newValidatorHandler**()

Creates a new ValidatorHandler for this Schema.

Returns:

Always return a non-null valid object.

15.2.3 Class SchemaFactory

java.lang.Object [SUN, 4, 2008]
└ **javax.xml.validation.SchemaFactory**

---

public abstract class **SchemaFactory**
extends Object

Factory that creates Schema objects. Entry-point to the validation API.

SchemaFactory is a schema compiler. It reads external representations of schemas and prepares them for validation.

The SchemaFactory class is not thread-safe. In other words, it is the application's responsibility to ensure that at most one thread is using a SchemaFactory object at any given moment. Implementations are encouraged to mark methods as synchronized to protect themselves from broken clients.

SchemaFactory is not re-entrant. While one of the newSchema methods is being invoked, applications may not attempt to recursively invoke the newSchema method, even from the same thread.

Schema Language

This spec uses a namespace URI to designate a schema language. The following table shows the values defined by this specification.

To be compliant with the spec, the implementation is only required to support W3C XML Schema 1.0. However, if it chooses to support other schema languages listed here, it must conform to the relevant behaviors described in this spec.

Schema languages not listed here are expected to introduce their own URIs to represent themselves. The SchemaFactory class is capable of locating other implementations for other schema languages at run-time.

Note that because the XML DTD is strongly tied to the parsing process and has a significant effect on the parsing process, it is impossible to define the DTD validation as a process independent from parsing. For this reason, this specification does not define the semantics for the XML DTD. This doesn't prohibit implentors from implementing it in a way they see fit, but *users are warned that any DTD validation implemented on this interface necessarily deviate from the XML DTD semantics as defined in the XML 1.0*.

value	language
XMLConstants.W3C_XML_SCHEMA_NS_URI ("http://www.w3.org/2001/XMLSchema")	W3C XML Schema 1.0
XMLConstants.RELAXNG_NS_URI ("http://relaxng.org/ns/structure/1.0")	RELAX NG 1.0

Since:

1.5

Constructor Summary	
protected	SchemaFactory() Constructor for derived classes.

Method Summary	
abstract ErrorHandler	getErrorHandler() Gets the current ErrorHandler set to this SchemaFactory.
boolean	getFeature(String name) Look up the value of a feature flag.

Method Summary	
Object	getProperty(String name) Look up the value of a property.
abstract LSResourceResolver	getResourceResolver() Gets the current LSResourceResolver set to this SchemaFactory.
abstract boolean	isSchemaLanguageSupported(String schemaLanguage) Is specified schema supported by this SchemaFactory?
static SchemaFactory	newInstance(String schemaLanguage) Lookup an implementation of the SchemaFactory that supports the specified schema language and return it.
abstract Schema	newSchema() Creates a special Schema object.
Schema	newSchema(File schema) Parses the specified File as a schema and returns it as a Schema.
Schema	newSchema(Source schema) Parses the specified source as a schema and returns it as a schema.
abstract Schema	newSchema(Source[] schemas) Parses the specified source(s) as a schema and returns it as a schema.
Schema	newSchema(URL schema) Parses the specified URL as a schema and returns it as a Schema.
abstract void	setErrorHandler(ErrorHandler errorHandler) Sets the ErrorHandler to receive errors encountered during the newSchema method invocation.
void	setFeature(String name, boolean value) Set the value of a feature flag.
void	setProperty(String name, Object object) Set the value of a property.
abstract void	setResourceResolver(LSResourceResolver resourceResolver) Sets the LSResourceResolver to customize resource resolution when parsing schemas.

15.2.4 Class StreamResult

java.lang.Object [SUN, 5, 2008]

└ **javax.xml.transform.stream.StreamResult**

All Implemented Interfaces:

Result

---

public class **StreamResult**

extends Object

implements Result

Acts as an holder for a transformation result, which may be XML, plain Text, HTML, or some other form of markup.

Field Summary	
static String	FEATURE If TransformerFactory.getFeature(java.lang.String) returns true when passed this value as an argument, the Transformer supports Result output of this type.

Fields inherited from interface javax.xml.transform.Result
PI_DISABLE_OUTPUT_ESCAPING, PI_ENABLE_OUTPUT_ESCAPING

Constructor Summary
StreamResult() Zero-argument default constructor.
StreamResult(File f) Construct a StreamResult from a File.
StreamResult(OutputStream outputStream) Construct a StreamResult from a byte stream.
StreamResult(String systemId) Construct a StreamResult from a URL.

Constructor Summary
StreamResult(Writer writer) Construct a StreamResult from a character stream.

Method Summary	
OutputStream	getOutputStream() Get the byte stream that was set with setOutputStream.
String	getSystemId() Get the system identifier that was set with setSystemId.
Writer	getWriter() Get the character stream that was set with setWriter.
void	setOutputStream(OutputStream outputStream) Set the ByteStream that is to be written to.
void	setSystemId(File f) Set the system ID from a File reference.
void	setSystemId(String systemId) Set the systemID that may be used in association with the byte or character stream, or, if neither is set, use this value as a writeable URI (probably a file name).
void	setWriter(Writer writer) Set the writer that is to receive the result.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

15.2.5 Class StreamSource

java.lang.Object [SUN, 6, 2008]

└ **javax.xml.transform.stream.StreamSource**

All Implemented Interfaces:

Source

public class **StreamSource**

extends Object

implements Source

Acts as an holder for a transformation Source in the form of a stream of XML markup.

Field Summary	
static String	FEATURE If TransformerFactory.getFeature(java.lang.String) returns true when passed this value as an argument, the Transformer supports Source input of this type.

Constructor Summary
StreamSource() Zero-argument default constructor.
StreamSource(File f) Construct a StreamSource from a File.
StreamSource(InputStream inputStream) Construct a StreamSource from a byte stream.
StreamSource(InputStream inputStream, String systemId) Construct a StreamSource from a byte stream.
StreamSource(Reader reader) Construct a StreamSource from a character reader.
StreamSource(Reader reader, String systemId) Construct a StreamSource from a character reader.
StreamSource(String systemId) Construct a StreamSource from a URL.

Method Summary	
InputStream	getInputStream() Get the byte stream that was set with setByteStream.

Method Summary	
String	getPublicId() Get the public identifier that was set with setPublicId.
Reader	getReader() Get the character stream that was set with setReader.
String	getSystemId() Get the system identifier that was set with setSystemId.
void	setInputStream(InputStream inputStream) Set the byte stream to be used as input.
void	setPublicId(String publicId) Set the public identifier for this Source.
void	setReader(Reader reader) Set the input to be a character reader.
void	setSystemId(File f) Set the system ID from a File reference.
void	setSystemId(String systemId) Set the system identifier for this Source.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

15.2.6 Class StringReader

java.lang.Object [SUN, 7, 2008]

└ java.io.Reader

└ **java.io.StringReader**

---

public class **StringReader**
extends Reader

A character stream whose source is a string.

Since:
JDK1.1

Field Summary

Fields inherited from class java.io.Reader
lock

Constructor Summary
StringReader(String s) Create a new string reader.

Method Summary	
void	close() Close the stream.
void	mark(int readAheadLimit) Mark the present position in the stream.
boolean	markSupported() Tell whether this stream supports the mark() operation, which it does.
int	read() Read a single character.
int	read(char[] cbuf, int off, int len) Read characters into a portion of an array.
boolean	ready() Tell whether this stream is ready to be read.
void	reset() Reset the stream to the most recent mark, or to the beginning of the string if it has never been marked.
long	skip(long ns) Skip characters.

Methods inherited from class java.io.Reader
read

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

15.2.7 Class StringWriter

java.lang.Object [SUN, 8, 2008]

└ java.io.Writer

└ **java.io.StringWriter**

All Implemented Interfaces:

Closeable, Flushable, Appendable

---

public class **StringWriter**

extends Writer

A character stream that collects its output in a string buffer, which can then be used to construct a string.

Closing a StringWriter has no effect. The methods in this class can be called after the stream has been closed without generating an IOException.

Since:

JDK1.1

Field Summary

Fields inherited from class java.io.Writer
lock

Constructor Summary
StringWriter() Create a new string writer, using the default initial string-buffer size.
StringWriter(int initialSize) Create a new string writer, using the specified initial string-buffer size.

Method Summary	
StringWriter	append(char c) Appends the specified character to this writer.
StringWriter	append(CharSequence csq) Appends the specified character sequence to this writer.
StringWriter	append(CharSequence csq, int start, int end) Appends a subsequence of the specified character sequence to this writer.
void	close() Closing a StringWriter has no effect.
void	flush() Flush the stream.
StringBuffer	getBuffer() Return the string buffer itself.
String	toString() Return the buffer's current value as a string.
void	write(char[] cbuf, int off, int len) Write a portion of an array of characters.
void	write(int c) Write a single character.
void	write(String str) Write a string.
void	write(String str, int off, int len) Write a portion of a string.

Methods inherited from class java.io.Writer
write

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

15.2.8 Class TeXFormula

java.lang.Object [SUN, 9, 2008]

└ **be.ugent.caagt.jmathtex.TeXFormula**

---

public class **TeXFormula**
extends java.lang.Object

Represents a logical mathematical formula that will be displayed (by creating a TeXIcon from it and painting it) using algorithms that are based on the TeX algorithms.

These formula's can be built using the built-in primitive TeX parser (methods with String arguments) or using other TeXFormula objects. Most methods have (an) equivalent(s) where one or more TeXFormula arguments are replaced with String arguments. These are just shorter notations, because all they do is parse the string(s) to TeXFormula's and call an equivalent method with (a) TeXFormula argument(s). Most methods also come in 2 variants. One kind will use this TeXFormula to build another mathematical construction and then change this object to represent the newly build construction. The other kind will only use other TeXFormula's (or parse strings), build a mathematical construction with them and insert this newly build construction at the end of this TeXFormula. Because all the provided methods return a pointer to this (modified) TeXFormula (except for the createTeXIcon method that returns a TeXIcon pointer), method chaining is also possible.

Important: All the provided methods modify this TeXFormula object, but all the TeXFormula arguments of these methods will remain unchanged and independent of this TeXFormula object!

Field Summary	
protected static float	PREC
protected be.ugent.caagt.jmathtex.Atom	root

Constructor Summary
TeXFormula() Creates an empty TeXFormula.
TeXFormula(java.util.List<TeXFormula> l) Creates a new TeXFormula from a list of TeXFormula objects.
TeXFormula(java.lang.String s) Creates a new TeXFormula by parsing the given string (using a primitive TeX parser).
TeXFormula(TeXFormula f) Creates a new TeXFormula that is a copy of the given TeXFormula.

Method Summary	
TeXFormula	add(java.lang.String s) Parses the given string and inserts the resulting formula at the end of the current TeXFormula.
TeXFormula	add(TeXFormula f) Inserts the given TeXFormula at the end of the current TeXFormula.
TeXFormula	addAcc(java.lang.String s, java.lang.String accentName) Parses the given string(s) into a TeXFormula, puts the given accent above it and inserts the result at the end of the current TeXFormula.
TeXFormula	addAcc(TeXFormula base, java.lang.String accentName) Puts the given accent above the given TeXFormula and inserts the result at the end of the current TeXFormula.
TeXFormula	addAcc(TeXFormula base, TeXFormula accent) Puts the given accent TeXFormula (that must represent a single accent symbol!) above the given base TeXFormula and inserts the result at the end of the current TeXFormula.

Method Summary	
TeXFormula	addEmbraced(java.lang.String s, char l, char r) Parses the given string into a TeXFormula, surrounds it with the given delimiters and inserts the result at the end of the current TeXformula.
TeXFormula	addEmbraced(java.lang.String s, java.lang.String left, java.lang.String right) Parses the given string(s) into a TeXFormula, surrounds it with the given delimiters (if not null) and inserts the result at the end of the current TeXFormula.
TeXFormula	addEmbraced(TeXFormula f, char l, char r) Surrounds the given TeXFormula with the given delimiters and inserts the result at the end of the current TeXformula.
TeXFormula	addEmbraced(TeXFormula f, java.lang.String left, java.lang.String right) Surrounds the given TeXFormula with the given delimiters (if not null) and inserts the result at the end of the current TeXFormula.
TeXFormula	addFraction(java.lang.String num, java.lang.String denom, boolean rule) Parses the given strings into TeXFormula's that will represent the numerator (num) and the denominator (denom) of a fraction, draws a line between them depending on "rule" and inserts the result at the end of the current TeXFormula.
TeXFormula	addFraction(java.lang.String num, java.lang.String denom, boolean rule, int numAlign, int denomAlign) Parses the given strings into TeXFormula's that will represent the numerator (num) and denominator (denom) of a fraction, draws a line between them depending on "rule", aligns the numerator and denominator in comparison with each other (indicated by numAlign and denomAlign) and inserts the result at the end of the current TeXFormula.
TeXFormula	addFraction(java.lang.String num, TeXFormula denom, boolean rule) Parses the given string into a TeXFormula that will represent the numerator of a fraction, uses the given TeXFormula as the denominator of this fraction, draws a line between them depending on "rule" and inserts the result at the end of the current TeXFormula.

Method Summary	
TeXFormula	addFraction(TeXFormula num, java.lang.String denom, boolean rule) Parses the given string into a TeXFormula that will represent the denominator of a fraction, uses the given TeXFormula as the numerator of this fraction, draws a line between them depending on "rule" and inserts the result at the end of the current TeXFormula.
TeXFormula	addFraction(TeXFormula num, TeXFormula denom, boolean rule) Uses the given TeXFormula's as the numerator (num) and denominator (denom) of a fraction, draws a line between them depending on "rule" and inserts the result at the end of the current TeXFormula.
TeXFormula	addFraction(TeXFormula num, TeXFormula denom, boolean rule, int numAlign, int denomAlign) Uses the given TeXFormula's as the numerator (num) and denominator (denom) of a fraction, draws a line between them depending on "rule", aligns the numerator and denominator in comparison with each other (indicated by numAlign and denomAlign) and inserts the result at the end of the current TeXFormula.
TeXFormula	addNthRoot(java.lang.String base, java.lang.String nthRoot) Parses the given strings into TeXFormula's, puts them under a root sign (base) and in the upper left corner over this root sign (nthRoot) and inserts the result at the end of the current TeXFormula.
TeXFormula	addNthRoot(java.lang.String base, TeXFormula nthRoot) Parses the given string into a TeXFormula, puts it under a root sign, puts the given TeXFormula in the upper left corner over this root sign and inserts the result at the end of the current TeXFormula.
TeXFormula	addNthRoot(TeXFormula base, java.lang.String nthRoot) Parses the given string into a TeXFormula, puts it in the upper left corner over the root sign, puts the given TeXFormula under this root sign and inserts the result at the end of the current TeXFormula.
TeXFormula	addNthRoot(TeXFormula base, TeXFormula nthRoot) Puts the given TeXFormula's under a root sign (base) and in the upper left corner over this root sign (nthRoot) and inserts the result at the end of the current TeXFormula.

Method Summary	
TeXFormula	addOp(java.lang.String op, java.lang.String low, java.lang.String up) Parses the given strings into TeXFormula's that will represent a "big operator" (op), it's lower (low) and upper (up) bound, and inserts the result at the end of the current TeXFormula.
TeXFormula	addOp(java.lang.String op, java.lang.String low, java.lang.String up, boolean lim) Parses the given strings into TeXFormula's that will represent a "big operator" (op), it's lower (low) and upper (up) bound, and inserts the result at the end of the current TeXFormula.
TeXFormula	addOp(TeXFormula op, TeXFormula low, TeXFormula up) Uses the given TeXFormula's as a "big operator" (op), it's lower (low) and upper (up) bound, and inserts the result at the end of the current TeXFormula.
TeXFormula	addOp(TeXFormula op, TeXFormula low, TeXFormula up, boolean lim) Uses the given TeXFormula's as a "big operator" (op), it's lower (low) and upper (up) bound, and inserts the result at the end of the current TeXFormula.
TeXFormula	addPhantom(java.lang.String phantom) Parses the given string into a phantom TeXFormula and inserts the result at the end of the current TeXFormula.
TeXFormula	addPhantom(java.lang.String phantom, boolean width, boolean height, boolean depth) Parses the given string into a phantom TeXFormula and inserts the result at the end of the current TeXFormula.
TeXFormula	addPhantom(TeXFormula phantom) Inserts the given TeXFormula as a phantom TeXFormula at the end of the current TeXFormula.
TeXFormula	addPhantom(TeXFormula phantom, boolean width, boolean height, boolean depth) Inserts the given TeXFormula as a phantom TeXFormula at the end of the current TeXFormula.
TeXFormula	addSqrt(java.lang.String base) Parses the given string into a TeXFormula that will be displayed under a root sign and inserts the result at the end of the current TeXFormula.

Method Summary	
TeXFormula	addSqrt(TeXFormula base) Displays the given TeXFormula under a root sign and inserts the result at the end of the current TeXFormula.
TeXFormula	addStrut(int unit, float width, float height, float depth) Inserts a strut box (whitespace) with the given width, height and depth (in the given unit) at the end of the current TeXFormula.
TeXFormula	addStrut(int widthUnit, float width, int heightUnit, float height, int depthUnit, float depth) Inserts a strut box (whitespace) with the given width (in widthUnits), height (in heightUnits) and depth (in depthUnits) at the end of the current TeXFormula.
TeXFormula	addSymbol(java.lang.String name) Inserts the symbol with the given name at the end of the current TeXFormula.
TeXFormula	addSymbol(java.lang.String name, int type) Inserts the symbol with the given name at the end of the current TeXFormula as a symbol of the given symbol type.
TeXFormula	centerOnAxis() Centers the current TeXformula vertically on the axis (defined by the parameter "axisheight" in the resource "DefaultTeXFont.xml".
TeXIcon	createTeXIcon(int style, float size) Creates a TeXIcon from this TeXFormula using the default TeXFont in the given point size and starting from the given TeX style.
TeXFormula	embrace(char left, char right) Surrounds this TeXFormula with the given delimiters.
TeXFormula	embrace(java.lang.String left, java.lang.String right) Surrounds this TeXFormula with the given delimiters (if not null).
TeXFormula	fraction(java.lang.String s, boolean rule) Uses the current TeXFormula as the numerator of a fraction, parses the given string into a TeXFormula that will represent the denominator of the fraction, draws a line between them depending on "rule" and changes the current TeXFormula into this resulting fraction.

Method Summary	
TeXFormula	fraction(java.lang.String s, boolean rule, int numAlign, int denomAlign) Uses the current TeXFormula as the numerator of a fraction, parses the given string into a TeXFormula that will represent the denominator of the fraction, possibly draws a line between them depending on "rule", aligns the numerator and denominator in comparison with each other (indicated by numAlign and denomAlign) and changes the current TeXFormula into this resulting fraction.
TeXFormula	fraction(TeXFormula f, boolean rule) Uses the current TeXFormula as the numerator of a fraction, the given TeXFormula as the denominator of the fraction, draws a line between them depending on "rule" and changes the current TeXFormula into this resulting fraction.
TeXFormula	fraction(TeXFormula f, boolean rule, int numAlign, int denomAlign) Uses the current TeXFormula as the numerator of a fraction, the given TeXFormula as the denominator of the fraction, draws a line between them depending on "rule", aligns the numerator and denominator in comparison with each other (indicated by numAlign and denomAlign) and changes the current TeXFormula into this resulting fraction.
TeXFormula	fraction(TeXFormula f, float defaultFactor, int numAlign, int denomAlign) Uses the current TeXFormula as the numerator of a fraction, the given TeXFormula as the denominator of the fraction, draws a line between them with a thickness of "defaultFactor" times the default rule thickness, aligns the numerator and denominator in comparison with each other (indicated by numAlign and denomAlign) and changes the current TeXFormula into this resulting fraction.
TeXFormula	fraction(TeXFormula f, int unit, float thickness) Uses the current TeXFormula as the numerator of a fraction, the given TeXFormula as the denominator of the fraction, draws a line between them with the given thickness (in the given unit) and changes the current TeXFormula into this resulting fraction.
TeXFormula	fraction(TeXFormula f, int unit, float thickness, int numAlign, int denomAlign) Uses the current TeXFormula as the numerator of a

Method Summary	
	fraction, the given TeXFormula as the denominator of the fraction, draws a line between them depending on "rule", aligns the numerator and denominator in comparison with each other (indicated by numAlign and denomAlign) and changes the current TeXFormula into this resulting fraction.
TeXFormula	fractionInvert(java.lang.String s, boolean rule) Uses the current TeXFormula as the denominator of a fraction, parses the given string into a TeXFormula that will represent the numerator of the fraction, draws a line between them depending on "rule" and changes the current TeXFormula into this resulting fraction.
TeXFormula	fractionInvert(java.lang.String s, boolean rule, int numAlign, int denomAlign) Uses the current TeXFormula as the denominator of a fraction, parses the given string into a TeXFormula that will represent the numerator of the fraction, draws a line between them depending on "rule", aligns the numerator and denominator in comparison with each other (indicated by numAlign and denomAlign) and changes the current TeXFormula into this resulting fraction.
TeXFormula	fractionInvert(TeXFormula f, boolean rule) Uses the current TeXFormula as the denominator of a fraction, the given TeXFormula as the numerator of the fraction, draws a line between them depending on "rule" and changes the current TeXFormula into this resulting fraction.
TeXFormula	fractionInvert(TeXFormula f, boolean rule, int numAlign, int denomAlign) Uses the current TeXFormula as the denominator of a fraction, the given TeXFormula as the numerator of the fraction, draws a line between them depending on "rule", aligns the numerator and denominator in comparison with each other (indicated by numAlign and denomAlign) and changes the current TeXFormula into this resulting fraction.
static TeXFormula	get(java.lang.String name) Get a predefined TeXFormula.
TeXFormula	makePhantom() Changes this TeXFormula into a phantom TeXFormula.
TeXFormula	makePhantom(boolean width, boolean height, boolean depth) Changes this TeXFormula into a phantom TeXFormula.

Method Summary	
TeXFormula	nthRoot(java.lang.String nthRoot) Parses the given string into a TeXFormula, puts it in the upper left corner over a root sign (nthRoot), puts the current TeXFormula under this root sign and changes the current TeXFormula into this resulting root construction.
TeXFormula	nthRoot(TeXFormula nthRoot) Puts the given TeXFormula in the upper left corner over a root sign, puts the current TeXFormula under this root sign and changes the current TeXFormula into this resulting root construction.
TeXFormula	overline() Puts a line over the current TeXFormula and changes the current TeXFormula into the resulting construction.
TeXFormula	putAccentOver(java.lang.String accentName) Puts the given accent above the current TeXFormula and changes the current TeXFormula into the resulting accent construction.
TeXFormula	putDelimiterOver(int delimiter) Puts the delimiter symbol represented by the given delimiter type constant above the current TeXFormula and changes the current TeXFormula into the resulting construction.
TeXFormula	putDelimiterOver(int delimiter, java.lang.String sup, int kernUnit, float kern) Puts the delimiter symbol represented by the given delimiter type constant above the current TeXFormula, parses the given string into a TeXFormula and puts it above the delimiter symbol (seperated by an amount of vertical space defined by the given float value and unit) in a smaller size (unless the current TeXFormula will be displayed in the smallest possible size: the script_script style's size) and finally changes the current TeXFormula into the resulting construction.
TeXFormula	putDelimiterOver(int delimiter, TeXFormula sup, int kernUnit, float kern) Puts the delimiter symbol represented by the given delimiter type constant above the current TeXFormula, puts the given TeXFormula above the delimiter symbol (seperated by an amount of vertical space defined by the given float value and

Method Summary	
	unit) in a smaller size (unless the current TeXFormula will be displayed in the smallest possible size: the "script_script" style's size) and finally changes the current TeXFormula into the resulting construction.
TeXFormula	putDelimiterUnder(int delimiter) Puts the delimiter symbol represented by the given delimiter type constant under the current TeXFormula and changes the current TeXFormula into the resulting construction.
TeXFormula	putDelimiterUnder(int delimiter, java.lang.String sub, int kernUnit, float kern) Puts the delimiter symbol represented by the given delimiter type constant under the current TeXFormula, parses the given string into a TeXFormula and puts it under the delimiter symbol (seperated by an amount of vertical space defined by the given float value and unit) in a smaller size (unless the current TeXFormula will be displayed in the smallest possible size: the script_script style's size) and finally changes the current TeXFormula into the resulting construction.
TeXFormula	putDelimiterUnder(int delimiter, TeXFormula sub, int kernUnit, float kern) Puts the delimiter symbol represented by the given delimiter type constant under the current TeXFormula, puts the given TeXFormula under the delimiter symbol (seperated by an amount of vertical space defined by the given float value and unit) in a smaller size (unless the current TeXFormula will be displayed in the smallest possible size: the "script_script" style's size) and finally changes the current TeXFormula into the resulting construction.
TeXFormula	putOver(java.lang.String over, int overUnit, float overSpace, boolean overScriptSize) Parses the given string into a TeXFormula that will be put above the current TeXFormula, in a smaller size depending on "overScriptSize" and seperated by a vertical space of size "overSpace" in "overUnit" and changes the current TeXFormula into the resulting construction.
TeXFormula	putOver(TeXFormula over, int overUnit, float overSpace, boolean overScriptSize) Puts the given TeXFormula above the current TeXFormula, in a smaller size depending on "overScriptSize"

Method Summary	
	and seperated by a vertical space of size "overSpace" in "overUnit" and changes the current TeXFormula into the resulting construction.
TeXFormula	putUnder(java.lang.String under, int underUnit, float underSpace, boolean underScriptSize) Parses the given string into a TeXFormula that will be put under the current TeXFormula, in a smaller size depending on "underScriptSize" and seperated by a vertical space of size "underSpace" in "underUnit" and changes the current TeXFormula into the resulting construction.
TeXFormula	putUnder(TeXFormula under, int underUnit, float underSpace, boolean underScriptSize) Puts the given TeXFormula under the current TeXFormula, in a smaller size depending on "underScriptSize" and seperated by a vertical space of size "underSpace" in "underUnit" and changes the current TeXFormula into the resulting construction.
TeXFormula	putUnderAndOver(java.lang.String under, int underUnit, float underSpace, boolean underScriptSize, java.lang.String over, int overUnit, float overSpace, boolean overScriptSize) Parses the given string "under" into a TeXFormula that will be put under the current TeXFormula, in a smaller size depending on "underScriptSize" and seperated by a vertical space of size "underSpace" in "underUnit", parses the given string "over" into a TeXFormula that will be put above the current TeXFormula, in a smaller size depending on "overScriptSize" and seperated by a vertical space of size "overSpace" in "overUnit" and finally changes the current TeXFormula into the resulting construction.
TeXFormula	putUnderAndOver(TeXFormula under, int underUnit, float underSpace, boolean underScriptSize, TeXFormula over, int overUnit, float overSpace, boolean overScriptSize) Puts the given TeXFormula "under" under the current TeXFormula, in a smaller size depending on "underScriptSize" and seperated by a vertical space of size "underSpace" in "underUnit", puts the given TeXFormula "over" above the current TeXFormula, in a smaller size depending on "overScriptSize" and seperated by a vertical space of size "overSpace" in "overUnit" and finally changes the current TeXFormula into the resulting construction.

Method Summary	
TeXFormula	setBackground(java.awt.Color c) Changes the background color of the current TeXFormula into the given color.
TeXFormula	setColor(java.awt.Color c) Changes the (foreground) color of the current TeXFormula into the given color.
TeXFormula	setFixedTypes(int leftType, int rightType) Sets a fixed left and right type of the current TeXFormula.
TeXFormula	setScripts(java.lang.String sub, java.lang.String sup) Parses the given strings into TeXFormula's and attaches them together to the current TeXFormula as a subscript (sub) and a superscript (sup).
TeXFormula	setScripts(java.lang.String sub, TeXFormula sup) Parses the given string into a TeXFormula's and attaches it to the current TeXFormula as a subscript together with the given TeXFormula (as a superscript).
TeXFormula	setScripts(TeXFormula sub, java.lang.String sup) Parses the given string into a TeXFormula's and attaches it to the current TeXFormula as a superscript together with the given TeXFormula (as a subscript).
TeXFormula	setScripts(TeXFormula sub, TeXFormula sup) Attaches the given TeXFormula's together to the current TeXFormula as a subscript (sub) and a superscript (sup).
TeXFormula	setSubscript(java.lang.String sub) Parses the given string into a TeXFormula and attaches it to the current TeXFormula as a subscript.
TeXFormula	setSubscript(TeXFormula sub) Attaches the given TeXFormula to the current TeXFormula as a subscript.
TeXFormula	setSuperscript(java.lang.String sup) Parses the given string into a TeXFormula and attaches it to the current TeXFormula as a superscript.
TeXFormula	setSuperscript(TeXFormula sup) Attaches the given TeXFormula to the current TeXFormula as a superscript.

Method Summary	
TeXFormula	sqrt() Puts the current TeXFormula under a root sign and changes the current TeXFormula into the resulting square root construction.
TeXFormula	underline() Puts a line under the current TeXFormula and changes the current TeXFormula into the resulting construction.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

15.2.9 Class TransformerFactory

java.lang.Object [SUN, 10, 2008]

└ **javax.xml.transform.TransformerFactory**

Direct Known Subclasses:

SAXTransformerFactory

---

public abstract class **TransformerFactory**

extends Object

A TransformerFactory instance can be used to create Transformer and Templates objects.

The system property that determines which Factory implementation to create is named "javax.xml.transform.TransformerFactory". This property names a concrete subclass of the TransformerFactory abstract class. If the property is not defined, a platform default is be used.

An implementation of the TransformerFactory class is *NOT* guaranteed to be thread safe. It is up to the user application to make sure about the use of the TransformerFactory from more than one thread. Alternatively the application can have one instance of the TransformerFactory per thread. An application can use the same instance of the factory to obtain one or more instances of a Transformer or Templates provided the instance of the factory isn't being used in more than one thread at a time.

Constructor Summary	
protected	TransformerFactory() Default constructor is protected on purpose.

Method Summary	
abstract Source	getAssociatedStylesheet(Source source, String media, String title, String charset) Get the stylesheet specification(s) associated via the xml-stylesheet processing instruction (see http://www.w3.org/TR/xml-stylesheet/) with the document document specified in the source parameter, and that match the given criteria.
abstract Object	getAttribute(String name) Allows the user to retrieve specific attributes on the underlying implementation.
abstract ErrorListener	getErrorListener() Get the error event handler for the TransformerFactory.
abstract boolean	getFeature(String name) Look up the value of a feature.
abstract URIResolver	getURIResolver() Get the object that is used by default during the transformation to resolve URIs used in document(), xsl:import, or xsl:include.
static TransformerFactory	newInstance() Obtain a new instance of a TransformerFactory.

abstract Templates	newTemplates(Source source) Process the Source into a Templates object, which is a a compiled representation of the source.
abstract Transformer	newTransformer() Create a new Transformer object that performs a copy of the source to the result.
abstract Transformer	newTransformer(Source source) Process the Source into a Transformer object.
abstract void	setAttribute(String name, Object value) Allows the user to set specific attributes on the underlying implementation.
abstract void	setErrorListener(ErrorListener listener) Set the error event listener for the TransformerFactory, which is used for the processing of transformation instructions, and not for the transformation itself.
abstract void	setURIResolver(URIResolver resolver) Set an object that is used by default during the transformation to resolve URIs used in xsl:import, or xsl:include.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

15.2.10 Class Validator

java.lang.Object [SUN, 11, 2008]

└ **javax.xml.validation.Validator**

---

public abstract class **Validator**

extends Object

A processor that checks an XML document against Schema.

A validator is a thread-unsafe and non-reentrant object. In other words, it is the application's responsibility to make sure that one Validator object is not used from more than one thread at any given time, and while the validate method is invoked, applications may not recursively call the validate method.

Note that while the validate(javax.xml.transform.Source) and validate(javax.xml.transform.Source, javax.xml.transform.Result) methods take a Source instance, the Source instance must be a SAXSource or DOMSource.

Since:

1.5

Constructor Summary	
protected	Validator() Constructor for derived classes.

Method Summary	
abstract ErrorHandler	getErrorHandler() Gets the current ErrorHandler set to this Validator.
boolean	getFeature(String name) Look up the value of a feature flag.
Object	getProperty(String name) Look up the value of a property.
abstract LSResourceResolver	getResourceResolver() Gets the current LSResourceResolver set to this Validator.
abstract void	reset() Reset this Validator to its original configuration.
abstract void	setErrorHandler(ErrorHandler errorHandler) Sets the ErrorHandler to receive errors encountered during the validate method invocation.

Method Summary	
void	setFeature(String name, boolean value) Set the value of a feature flag.
void	setProperty(String name, Object object) Set the value of a property.
abstract void	setResourceResolver(LSResourceResolver resourceResolver) Sets the LSResourceResolver to customize resource resolution while in a validation episode.
void	validate(Source source) Validates the specified input.
abstract void	validate(Source source, Result result) Validates the specified input and send the augmented validation result to the specified output.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

15.2.11 Interface Source

All Known Implementing Classes: [SUN, 12, 2008]

DOMSource, SAXSource, StreamSource

---

public interface **Source**

An object that implements this interface contains the information needed to act as source input (XML source or transformation instructions).

Method Summary	
String	getSystemId() Get the system identifier that was set with setSystemId.

Method Summary	
void	setSystemId(String systemId) Set the system identifier for this Source.

Method Detail

setSystemId

public void **setSystemId**(String systemId)

Set the system identifier for this Source.

The system identifier is optional if the source does not get its data from a URL, but it may still be useful to provide one. The application can use a system identifier, for example, to resolve relative URIs and to include in error messages and warnings.

Parameters:

systemId - The system identifier as a URL string.

---

getSystemId

public String **getSystemId**()

Get the system identifier that was set with setSystemId.

Returns:

The system identifier that was set with setSystemId, or null if setSystemId was not called.

15.2.12 Class Transformer

java.lang.Object [SUN, 13, 2008]

└ **javax.xml.transform.Transformer**

---

public abstract class **Transformer**
extends Object

An instance of this abstract class can transform a source tree into a result tree.

An instance of this class can be obtained with the TransformerFactory.newTransformer method. This instance may then be used to process XML from a variety of sources and write the transformation output to a variety of sinks.

An object of this class may not be used in multiple threads running concurrently. Different Transformers may be used concurrently by different threads.

A Transformer may be used multiple times. Parameters and output properties are preserved across transformations.

Constructor Summary	
protected	Transformer() Default constructor is protected on purpose.

Method Summary	
abstract void	clearParameters() Clear all parameters set with setParameter.
abstract ErrorListener	getErrorListener() Get the error event handler in effect for the transformation.
abstract Properties	getOutputProperties() Get a copy of the output properties for the transformation.
abstract String	getOutputProperty(String name) Get an output property that is in effect for the transformation.

Method Summary	
abstract Object	getParameter(String name) Get a parameter that was explicitly set with setParameter or setParameters.
abstract URIResolver	getURIResolver() Get an object that will be used to resolve URIs used in document(), etc.
abstract void	setErrorListener(ErrorListener listener) Set the error event listener in effect for the transformation.
abstract void	setOutputProperties(Properties oformat) Set the output properties for the transformation.
abstract void	setOutputProperty(String name, String value) Set an output property that will be in effect for the transformation.
abstract void	setParameter(String name, Object value) Add a parameter for the transformation.
abstract void	setURIResolver(URIResolver resolver) Set an object that will be used to resolve URIs used in document().
abstract void	transform(Source xmlSource, Result outputTarget) Process the source tree to the output result.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

15.2.13 Interface Result

All Known Implementing Classes:

DOMResult, SAXResult, StreamResult

---

public interface **Result**

An object that implements this interface contains the information needed to build a transformation result tree.

Field Summary	
static String	PI_DISABLE_OUTPUT_ESCAPING The name of the processing instruction that is sent if the result tree disables output escaping.
static String	PI_ENABLE_OUTPUT_ESCAPING The name of the processing instruction that is sent if the result tree enables output escaping at some point after having received a PI_DISABLE_OUTPUT_ESCAPING processing instruction.

Method Summary	
String	getSystemId() Get the system identifier that was set with setSystemId.
void	setSystemId(String systemId) Set the system identifier for this Result.

Field Detail

PI_DISABLE_OUTPUT_ESCAPING

public static final String **PI_DISABLE_OUTPUT_ESCAPING**

The name of the processing instruction that is sent if the result tree disables output escaping.

Normally, result tree serialization escapes & and < (and possibly other characters) when outputting text nodes. This ensures that the output is well-formed XML. However, it is sometimes convenient to be able to produce output that is almost, but not quite well-formed XML; for example, the output may include ill-formed sections that will be transformed into well-formed XML by a subsequent non-XML aware process. If a processing instruction is sent with this name, serialization should be output without any escaping.

Result DOM trees may also have PI_DISABLE_OUTPUT_ESCAPING and PI_ENABLE_OUTPUT_ESCAPING inserted into the tree.

See Also:

disable-output-escaping in XSLT Specification, Constant Field Values

---

PI_ENABLE_OUTPUT_ESCAPING

public static final String **PI_ENABLE_OUTPUT_ESCAPING**

The name of the processing instruction that is sent if the result tree enables output escaping at some point after having received a PI_DISABLE_ OUTPUT_ESCAPING processing instruction.

See Also:

disable-output-escaping in XSLT Specification, Constant Field Values

Method Detail

setSystemId

public void **setSystemId**(String systemId)

Set the system identifier for this Result.

If the Result is not to be written to a file, the system identifier is optional. The application may still want to provide one, however, for use in error messages and warnings, or to resolve relative output identifiers.

Parameters:

systemId - The system identifier as a URI string.

---

getSystemId

public String **getSystemId**()

Get the system identifier that was set with setSystemId.

Returns:

The system identifier that was set with setSystemId, or null if setSystemId was not called.

15.2.14 Interface HelpBroker

All Known Implementing Classes:

DefaultHelpBroker

---

public interface HelpBroker

The HelpBroker is the default presentation of a HelpSet. A HelpBroker is an abstraction of the presentation for a HelpSet; a straight-forward implementation is a JHelp() on the HelpSet. A HelpBroker can be asked to show a given Navigational View, and can display a given ID (help topic).

See Also:

HelpSet, JHelpNavigator, javax.help.HelpVisitListener

Method Summary	
void	enableHelp(java.awt.Component comp, java.lang.String id, HelpSet hs) Enables help for a component.
void	enableHelp(java.awt.MenuItem comp, java.lang.String id, HelpSet hs) Enables help for a MenuItem.
void	enableHelpKey(java.awt.Component comp, java.lang.String id, HelpSet hs) Enables the Help key on a component.
void	enableHelpOnButton(java.awt.Component comp, java.lang.String id, HelpSet hs) Enables help for a component.
void	enableHelpOnButton(java.awt.MenuItem comp, java.lang.String id, HelpSet hs) Enables help for a MenuItem.

Map.ID	getCurrentID() Determines The currently displayed ID (if any).
java.net.URL	getCurrentURL() Determines the currently displayed ID.
java.lang.String	getCurrentView() Gets name of the current navigational view.
java.awt.Font	getFont() Gets the font for this HelpBroker.
HelpSet	getHelpSet() Gets the current HelpSet for this JavaHelp object.
java.util.Locale	getLocale() Returns the locale of this object.
java.awt.Point	getLocation() Gets the location of the presentation.
java.awt.Dimension	getSize() Gets the size of the presentation.
void	initPresentation() Initializes the presentation.
boolean	isDisplayed() Determines if the presentation is visible.
boolean	isViewDisplayed() Determines if the Navigational View is visible.
void	setCurrentID(Map.ID id) Displays this ID.
void	setCurrentID(java.lang.String id) Displays this ID.
void	setCurrentURL(java.net.URL url) Displays this ID.
void	setCurrentView(java.lang.String name) Activates the Navigator view with a given name.
void	setDisplayed(boolean displayed) Displays the presentation to the user.
void	setFont(java.awt.Font f) Sets the font for this HelpBroker.
void	setHelpSet(HelpSet hs) Sets the current HelpSet for this HelpBroker.

void	setLocale(java.util.Locale l) Sets the locale of this HelpBroker.
void	setLocation(java.awt.Point p) Sets the position of the presentation.
void	setSize(java.awt.Dimension d) Sets the size of the presentation.
void	setViewDisplayed(boolean displayed) Hides/Shows Navigational Views.

15.2.15 Class HelpSet

java.lang.Object

javax.help.HelpSet

---

public class HelpSet

extends java.lang.Object

A HelpSet is a collection of help information consisting of a HelpSet file, table of contents (TOC), index, topic files, and Map file. The HelpSet file is the portal to the HelpSet.

Nested Class Summary	
static class	HelpSet.DefaultHelpSetFactory The default HelpSetFactory that processes HelpSets.

Field Summary	
static java.lang.String	helpBrokerClass
static java.lang.String	helpBrokerLoader
static java.lang.Object	implRegistry Information for implementation customization.
static java.lang.Object	kitLoaderRegistry

static java.lang.Object	kitTypeRegistry HelpSet context information.
static java.lang.String	publicIDString PublicID (known to this XML processor) to the DTD for version 1.0 of the HelpSet

Constructor Summary
HelpSet() Creates an empty HelpSet.
HelpSet(java.lang.ClassLoader loader) Creates an empty HelpSet that one can parse into.
HelpSet(java.lang.ClassLoader loader, java.net.URL helpset) Creates a HelpSet.

Method Summary	
void	add(HelpSet hs) Adds a HelpSet, HelpSetEvents are generated.
void	addHelpSetListener(HelpSetListener l) Adds a listener for the HelpSetEvent posted after the model has changed.
boolean	contains(HelpSet hs) Determines if a HelpSet is a sub-HelpSet of this object.
HelpBroker	createHelpBroker() Creates a presentation object for this HelpSet.
static java.net.URL	findHelpSet(java.lang.ClassLoader cl, java.lang.String name) As above but default on locale to Locale.getDefault()
static java.net.URL	findHelpSet(java.lang.ClassLoader cl, java.lang.String name, java.util.Locale locale) Locates a HelpSet file and return its URL.
static java.net.URL	findHelpSet(java.lang.ClassLoader cl, java.lang.String shortName, java.lang.String extension, java.util.Locale locale) Locates a HelpSet file and return its URL.

Map	getCombinedMap() The map for this HelpSet.
java.util.Enumeration	getHelpSets() Enumerates all the HelpSets that have been added to this one.
java.net.URL	getHelpSetURL() The URL that is the base for this HelpSet.
Map.ID	getHomeID() Returns the ID to visit when the user makes a "go home" gesture.
java.lang.Object	getKeyData(java.lang.Object context, java.lang.String key) Gets some Data for a Key in a given context.
java.lang.ClassLoader	getLoader() A classloader to use when locating classes.
java.util.Locale	getLocale() Gets the locale for this HelpSet.
Map	getLocalMap() Get the local (i.e. non-recursive) Map for this HelpSet.
NavigatorView	getNavigatorView(java.lang.String name) Gets the NavigatorView with a specific name.
NavigatorView[]	getNavigatorViews() NavigatorView describes the navigator views that are requested by this HelpSet.
java.lang.String	getTitle() Gets the title of this HelpSet.
static HelpSet	parse(java.net.URL url, java.lang.ClassLoader loader, javax.help.HelpSetFactory factory) Parsed a HelpSet file.
void	parseInto(java.net.URL url, javax.help.HelpSetFactory factory) Parses into this HelpSet.
boolean	remove(HelpSet hs) Removes a HelpSet from this HelpSet; HelpSetEvents are generated Return True if it is found, otherwise false.
void	removeHelpSetListener(HelpSetListener l) Removes a listener previously added with

	addHelpSetListener
void	setHomeID(java.lang.String homeID) Sets the Home ID for a HelpSet.
void	setKeyData(java.lang.Object context, java.lang.String key, java.lang.Object data) Sets some local KeyData on a given context.
void	setLocalMap(Map map) Set the Map for this HelpSet.
void	setTitle(java.lang.String title) Sest the title for this HelpSet.
java.lang.String	toString() Prints Name for this HelpSet.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

15.2.16 Class ExampleFileFilter

java.lang.Object

javax.swing.filechooser.FileFilter

apollo.util.ExampleFileFilter

---

public class ExampleFileFilter

extends javax.swing.filechooser.FileFilter

A convenience implementation of FileFilter that filters out all files except for those type extensions that it knows about. Extensions are of the type ".foo", which is typically found on Windows and Unix boxes, but not on Macinthosh. Case is ignored. Example - create a new filter that filerts out all files but gif and jpg image files: JFileChooser chooser = new JFileChooser(); ExampleFileFilter filter = new ExampleFileFilter(new

String{"gif", "jpg"}, "JPEG & GIF Images")

chooser.addChoosableFileFilter(filter); chooser.showOpenDialog(this);

Version:

1.9 04/23/99

Author:

Jeff Dinkins

Constructor Summary
ExampleFileFilter() Creates a file filter.
ExampleFileFilter(java.lang.String extension) Creates a file filter that accepts files with the given extension.
ExampleFileFilter(java.lang.String[] filters) Creates a file filter from the given string array.
ExampleFileFilter(java.lang.String[] filters, java.lang.String description) Creates a file filter from the given string array and description.
ExampleFileFilter(java.lang.String extension, java.lang.String description) Creates a file filter that accepts the given file type.

Method Summary	
boolean	accept(java.io.File f) Return true if this file should be shown in the directory pane, false if it shouldn't.
void	addExtension(java.lang.String extension) Adds a filetype "dot" extension to filter against.
java.lang.String	getDescription() Returns the human readable description of this filter.
java.lang.String	getExtension(java.io.File f) Return the extension portion of the file's name .
boolean	isExtensionListInDescription() Returns whether the extension list (.jpg, .gif, etc) should show up in the human readable description.
void	setDescription(java.lang.String description) Sets the human readable description of this filter.

void	setExtensionListInDescription(boolean b) Determines whether the extension list (.jpg, .gif, etc) should show up in the human readable description.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

16 BIBLIOGRAFÍA

[ACSC, 1995] Asian Computer Science Conference ACSC'95. “Constraints for Free in Concurrent Computation”, 1995.

[ACSC, 1995] Asian Computer Science Conference ACSC'95. “Constraints for Free in Concurrent Computation” 1995.

[Aho et al., 1998] Alfred V. Aho, Ravi Sethi y Jeffrey D. Ullman. "Compiladores. Principios, técnicas y herramientas". Editorial Pearson - Addison Wesley Longman. Mexico, 1998.

[Bass et al., 1998] Len Bass, Paul Clemens, Rick Kazman, “Software Architecture in Practice”. Editorial Addison Wesley Longman Inc., The SEI series in Software Engineering, 1998.

[Benz y Durant, 2003] Brian Benz y Jhon Durant. “XML Programming Bible”. Editorial Wiley Publishing, Inc. Estados Unidos, 2003.

[Bergstra y Klop, 1985] Bergstra, J. A. y Klop, J. W. “Algebra of Communicating Processes with Abstraction”. Theoretical Computer Science. 1985.

[Buschmann et al., 1996] Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad y M. Stal. “Pattern-Oriented Software Architecture: A System Of Patterns”; West Sussex, England: John Wiley & Sons Ltd., 1996.

[Buschmann et al., 1996] Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad y M. Stal. “Pattern-Oriented Software Architecture: A System Of Patterns”; West Sussex, England: John Wiley & Sons Ltd., 1996.

[Canal, 2000] Carlos Canal Velasco, “Un lenguaje para la especificación y validación de arquitecturas de software”. Tesis doctoral de la Universidad de Malaga, España. 2000.

[Cernuda, 2004] Agustín Cernuda del Rio, “XSLT/XPath”. Departamento de Informática – Universidad de Oviedo. España, 2004.

[Chen y Norman, 1992] M. Chen y R. J. Norman, “A framework for integrated case, IEEE Software”, vol. 9, Marzo de 1992.

[Daza y Quiroga, 2004] Saulo David Daza y Juan Pablo Quiroga. “Proceso para la gestión de riesgos en los proyectos ágiles”. Universidad de los Andes, Bogotá (Colombia), 2004.

[Diosa et al., 2005] Henry Diosa, Carlos Gaona y Juan Díaz. “Cálculo para el Modelamiento Formal de Arquitecturas de Software Basadas en Componentes y con Restricciones en Tiempo de Ejecución”. Santiago de Cali (Colombia), Octubre de 2005.

[Diosa et al., 2005] Henry Diosa, Carlos Gaona y Juan Díaz. “Cálculo para el Modelamiento Formal de Arquitecturas de Software Basadas en Componentes y con

Restricciones en Tiempo de Ejecución". Santiago de Cali (Colombia), Octubre de 2005.

[Diosa, 2005] Diosa, Henry, "Cálculo formal para especificar aspectos dinámicos de arquitecturas de software con componentes condicionados", Reporte de Investigación RI-2005-02-VI, septiembre de 2005.

[Floyd, 2000] Floyd, Michael. "Creación de sitios WEB con XML". Editorial Prentice Hall. España, 2000.

[Fuentes et al., 2001] Lidia Fuentes, José M. Troya y Antonio Vallecillo, "Desarrollo de Software Basado en Componentes". Dept. Lenguajes y Ciencias de la Computación. Universidad de Málaga. 2001.

[Fuentes et al., 2001] Lidia Fuentes, José M. Troya y Antonio Vallecillo. "Desarrollo de Software Basado en Componentes". Dept. Lenguajes y Ciencias de la Computación. Universidad de Málaga. España, 2001.

[Gabrick y Weiss] Kurt A. Gabrick y David B. Weiss. "Java 2EE and XML development". Editorial Manning Publications Co. Estados Unidos, 2002.

[Hoare, 1985] Hoare, C. A. R. "Communicating Sequential Processes". Editorial Prentice Hall. 1985.

[Kruchten, 2001] Philippe Kruchten, "The Rational Unified Process An Introduction", Addison Wesley, 2001.

[Kruchten, 2001] Philippe Kruchten, "The Rational Unified Process An Introduction", Addison Wesley, 2001.

[Milner, 1989] Robin Milner. “Communication and Concurrency”. Editorial Prentice Hall. 1969.

[Milner, 1999] Robin Milner. “Communicating and Mobile Systems: The π calculus”. Cambridge University Press, 1999.

[NIST, 1991] NIST Special Publication 500-201. “NIST/ECMA reference model for frameworks of software engineering environments”. Diciembre de 1991.

[Paulk, 1985] M. C. Paulk, “The ARC Network: A case study, IEEE Software”, vol. 2, Mayo de 1985.

[Pressman, 2002] Roger Pessman. “Ingeniería de software un enfoque practico”, 5ª Edición. Editorial Mc Graw Hill, España, 2002.

[Pressman, 2002] Roger Pessman. “Ingeniería de software un enfoque practico”, 5ª Edición. Editorial Mc Graw Hill, España, 2002.

[Pressman, 2002] Roger Pessman. “Ingeniería de software un enfoque practico”, 5ª Edición. Editorial Mc Graw Hill, España, 2002.

[Reynoso y Kicillof, 2004] Carlos Reynoso y Nicolás Kicillof. “Lenguajes de Descripción de Arquitectura”. Universidad De Buenos Aires. 2004.

[Reynoso y Kicillof, 2004] Carlos Reynoso y Nicolás Kicillof. “Lenguajes de Descripción de Arquitectura”. Universidad De Buenos Aires. 2004.

[RSC, 2002] Rational Software Corporation, Product: Rational Software Corporation, 2002

[Scheifler y Gettys, 1986] R. W. Scheifler y J. Gettys, “The X window system,” AACM Transactions on Graphics, vol. 5, pp, Abril de 1986.

[Szyperski y Pfister, 1997] Szyperski, C. y Pfister, C. “Summary of the Workshop on Component Oriented Programming (WCOP’96).” Mühlhauser, M. (ed.), Special Issues in Object-Oriented Programming. Workshop Reader of ECOOP’96. Dpunkt Verlag. 1997.

[Szyperski, 1996] Szyperski, C. “Independently Extensible Systems —Software Engineering Potential and Challenges”—. Proc. of the 19th Australasian Computer Science Conference, Melbourne. 1996.

[Szyperski, 1998] Szyperski, C. “Component Software. Beyond Object-Oriented Programming”. Editorial Addison-Wesley. 1998.

[Szyperski, 1998] Szyperski, C. “Component Software. Beyond Object-Oriented Programming”. Editorial Addison-Wesley. 1998.

[Tucker, 2004] Allen B. Tucker. “Computer Science Handbook”, 2ª Edición. Editorial Routledge,Estados Unidos 2004.

[Vestal, 1993] Steve Vestal. “A cursory overview and comparison of four Architecture Description Languages”. Technical Report, Honeywell Technology Center. 1993.

[Vestal, 1993] Steve Vestal. “A cursory overview and comparison of four Architecture Description Languages”. Technical Report, Honeywell Technology Center. 1993.

17 REFERENCIAS PÁGINAS WEB ESPECIALIZADAS

[Armstrong, 2007] Armstrong, Eric. “Working with XML”. Citado: 26 julio de 2007. URL: http://java.sun.com/webservices/jaxp/dist/1.1/docs/tutorial/index.html

[Arquisoft, 2008] Universidad Distrital Francisco José de Caldas - Grupo de investigación – Arquitecturas de Software. “Traducción de especificaciones arquitecturales de software en xADL2.0 estereotipadas a formato de visualización factorial SVG por Ana Milena Parra y Juan Carlos Reyes”. Citado: 25 de enero de 2008. URL:http://www.udistrital.edu.co/comunidad/grupos/arquisoft/

[Cleemput et al., 2008] Kris Coolsaet, Nico Van Cleemput y Kurt Vermeulen. “JMathTeX”. Citado: 20 de Junio de 2008. URL: http://jmathtex.sourceforge.net/index.html.

[Dashofy, 2007] Eric M. Dashofy. “A Guide for Users of the xADL 2.0 Language”. Citado: 15 de agosto de 2007.

URL:
http://www.isr.uci.edu/projects/xarchuci/guide.html

[Eclipse, 2008] Eclipse Foundation. “Eclipse”. Citado: 15 de Junio de 2008. URL: http://www.eclipse.org/downloads/.

[ISR, 2007] Information Systems Research. “Schema types.xsd”. Citado: 15 de agosto de 2007. URL: http://www.isr.uci.edu/projects/xarchuci/xmlspy-docs/types/types.html#complexType_ArchTypes_Link0147D100

[ISR, 2008] Institute for Software Research (ISR) “xADL 2.0 SCHEMAS - xADL 2.0 Extensions Overview”-+. Citado: 20 de enero de 2008. URL: http://www.-isr.uci.edu/projects/xarchuci/ext-overview.html

[MSDN, 2007] Microsoft Developer Network. “Referencia de las normas XML”. Citado: 15 de agosto de 2007. URL: http://msdn2.microsoft.com/es-es/library/ms25605-8(VS.80).aspx

[Refsnes, 2007] Refsnes, Hege. “W3Schools”. Citado: 25 de julio de 2007. URL: http://www.w3schools.com/xml/xml_whatis.asp

[Reino, 2008] Alfredo Reino. “Introducción a las hojas de estilo XSL para XML”. Citado: 16 de enero de 2008. URL: http://www.webtaller.com/construccion/-lenguajes/xml/lecciones/introduccion-hojas-estilo-xsl-xml.php

[SUN, 1, 2008] Sun Microsystems. “Sun Developer Network SDN”. Citado: 15 de Junio de 2008. URL: http://java.sun.com/javase/downloads/?intcmp=1281.

[SUN, 2, 2008] Sun Microsystems. “JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Class File”. Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.4.2/docs/api/java/io/File.html

[SUN, 3, 2008] Sun Microsystems. “JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Class Schema”. Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/Schema.html

[SUN, 4, 2008] Sun Microsystems. “JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Class SchemaFactory”. Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/ SchemaFactory.html

[SUN, 5, 2008] Sun Microsystems. “JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Class StreamResult”. Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/StreamResult.html

[SUN, 6, 2008] Sun Microsystems. “JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Class

StreamSource". Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/StreamSource.html

[SUN, 7, 2008] Sun Microsystems. "JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Class StringReader". Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/ StringReader.html

[SUN, 8, 2008] Sun Microsystems. "JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Class StringWriter". Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/StringWriter.html

[SUN, 9, 2008] Sun Microsystems. "JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Class TeXFormula". Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/ TeXFormula.html

[SUN, 10, 2008] Sun Microsystems. "JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Class TransformerFactory". Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/TransformerFactory.html

[SUN, 11, 2008] Sun Microsystems. "JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Class Validator". Citado: 21 de abril de 2008. URL:

http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/ Validator.html

[SUN, 12, 2008] Sun Microsystems. “JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Interface Source”. Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.4.2/docs/api/javax/xml/transform/Source.html

[SUN, 13, 2008] Sun Microsystems. “JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification – Interface Source”. Citado: 21 de abril de 2008. URL: http://java.sun.com/j2se/1.4.2/docs/api/javax/xml/transform/Transformer.html

[TEK, 2008] Teknoda. “Notas Tecnicas de JAVA”. Citado: 1 de noviembre de 2008. URL: http://www.teknoda.com.ar/tips/java/tipja07.pdf

[UCI, 2007] Universidad de California, Irvine. “ArchStudio4 Software and Systems Architecture Development Enviroment”. Citado: 30 de julio de 2007. URL: http://www.isr.uci.edu/projects/archstudio/setup-fromsource.html

[Urretavizcaya et al., 2007] Ibon Urretavizcaya y Gorka Olaizola. “XSLT mini COMO”. Citado: 15 de agosto de 2007. URL: http://xml.utilitas.org/xslt_mini_como.html#i__1168

[W3C, 2008] World Wide Web Consortium. “Canonical XML Version 1.0”. Citado: 15 de Junio de 2008. URL: http://www.w3.org/TR/xml-c14n

[W3C, 1, 2007] World Wide Web Consortium (W3C). “XML Schema”. Citado: 14 de agosto de 2007. URL: http://www.w3.org/XML/Schema.

[W3C, 1, 2008] World Wide Web Consortium (W3C). “CSS & XSL”. Citado: 14 de enero de 2008. URL: http://www.w3.org/Style/CSS-vs-XSL

[W3C, 2, 2007] World Wide Web Consortium (W3C). “Extensible Markup Language (XML) 1.1 (Second Edition)”. Citado: 11 de septiembre de 2007. URL: http://www.w3.org/TR/xml11/.