Universidad Católica de Santa María

Escuela de Post Grado

Maestría en Ingeniería de Software

Exposición del Modelo Transaccional COM+ Para su Utilización en Actividades Transaccionales Todo o Nada Basadas en Servicios Web

Presentado por el Bachiller:

Hugo Christian Montenegro Beltrán

Para optar el Grado Académico de

Magíster en Ingeniería de Software

Arequipa – marzo de 2007

DEDICATORIA

Tu me das paz y alegría, transformando mi noche en día.

Gracias Glenycita, esposa mía, por apoyarme en este nuevo peldaño.

A mis papas Hugo – Nelly y mis hermanas Jessica - Paola, ustedes son mi inspiración para librar la gran batalla.

Gracias Jesús tu que siempre nos guías en nuestro camino, que se cumpla tu palabra sobre todas las cosas. Amén

"El hombre debe ser siempre flexible como la caña, no rígido como el cedro."

(Johann J. Engel)

INDICE GENERAL

LISTA DE FIGURAS

LISTA DE CUADROS

LISTA DE GRÁFICOS

RESUMEN

La capacidad de desarrollo de aplicaciones distribuidas que caracteriza al modelo de los servicios Web es realmente sorprendente. Por ejemplo, una empresa puede tener un servicio de pago electrónico en línea y ofrecérselo a sus socios que, a su vez, pueden conectarse a el independientemente de la plataforma que utilicen. Las empresas de alquiler de autos pueden vincular sus sistemas de reservas en línea con líneas aéreas y hoteles, con el fin de que el cliente pueda reservar un auto, un vuelo, y una habitación de hotel a la vez. A medida que empresas de envíos, de servicios y de pago electrónico comiencen a ofrecer sus sistemas por medio de los servicios Web, se facilitará la conexión a los sitios de comercio electrónico que se estén creando.

Los servicios Web aportan una solución a la necesidad de interoperabilidad entre aplicaciones Web al especificar un protocolo estándar de base XML. Al considerar los servicios Web como una nueva posibilidad de desarrollo de aplicaciones, los sitios expuestos por los Servicios Web deben considerarse como funciones (es decir, servicios Web). Una aplicación Web llama a otra, de la misma manera que una aplicación normal invoca una función y obtiene un resultado como respuesta. En el caso de que la comunicación sea de tipo asíncrona, una aplicación Web envía un mensaje a otra aplicación Web sin la necesidad de recibir una respuesta inmediata. Al contar con este tipo de interfaz de programas, una empresa puede centrar su atención en sus capacidades básicas dentro de su aplicación Web y añadir aquellas que

necesite de otros. De esta manera, el usuario consigue una aplicación completa a pesar de que un único proveedor solamente proporciona parte de la funcionalidad total.

La Tecnología COM+ permite el desarrollo de componentes reutilizables en base a una infraestructura que provee varios servicios para el desarrollador, los cuales no tienen que ser implementados para cada desarrollo, simplemente se tiene acceso a ellos cuando se desarrollan componentes bajo este marco de trabajo, los servicios más importantes son: manejo de transacciones, escalabilidad , seguridad .

Las características técnicas mencionadas anteriormente plantean nuevos retos en el área del diseño y desarrollo de los servicios Web, en especial para aquellos servicios web que desean exponer funcionalidad ya implementada en componentes COM+, esto permitirá el ahorro de tiempo y dinero en la implementación de nuevos sistemas que reutilicen la funcionalidad de componentes ya existentes, hasta el momento no hay investigaciones acerca de la utilización de componentes COM+ en actividades transaccionales basadas en servicios web.

En esta tesis se describe el análisis y diseño de una propuesta que permitirá participar a los componentes COM+ en actividades transaccionales basadas en servicios web. Para validar el diseño se harán uso de patrones de diseño de software, de tal manera que se asegure que las mejores prácticas de ingeniería

de software han sido cumplidas, el diseño además será probado con una implementación de un aplicativo, el uso y adecuación de las características transaccionales de los componentes COM+ serán evaluadas por un conjunto de expertos locales en el uso de esta tecnología.

ABSTRACT

The capacity of distributed applications development that characterizes the model of the Web services is really surprising. For example, a company could have an electronic payment service on line and offering it to its partners which, as well, can connect themselves independently of the platform they use. The companies of car rental can tie their reservations systems on line with air lines and hotels, in order that the client can reserve a car, a flight, and a hotel room simultaneously. As companies of shipments, services and electronic payment begin to offer their systems by means of the Web services, the connection to the sites of electronic commerce will be facilitated to those that are being created.

The Web services contribute to solve the necessity of interoperability between Web applications when specifying a standard protocol XML base. When considering the Web services like a new development possibility of applications, the sites exposed by the Web services must be considered like functions (It means, Web services). An application Web calls another one, in the same way that a normal application invokes a function and it obtains a result like answer. In case the communication is asynchronous, a Web application sends a message to another Web application without the necessity of receiving an immediate answer. When counting on this type of program's interface, a company can focus its attention on its basic capacities in its Web application

and to add those that others need. This way, the user obtains a complete application although an only supplier only provides part of the total functionality.

Technology COM+ allows the development of re-usable components on the basis of an infrastructure that provides several services for the developer, which do not have to be implemented for each development, it's just necessary to have access to them when components under this framework are developed, the most important services are: transactions handling, scalability, security.

The mentioned technical characteristics previously raise new challenges in the area of the design and development of the Web services, in special for those Web services that wish to expose already implemented functionality in COM+ components, this will allow save time and money in the implementation of new systems which re-use the already existing components functionality, until the moment there are not investigations about the use of components COM+ in transactional activities based on Web services.

This thesis describe the analysis and design of a proposal that will allow participate COM+ components in transactional activities based on Web services. To validate the design it will be use software design patrons, so it will be ensure that the software engineering best practices have been kept, besides the design will be proved with an application implementation, the COM+ components transactional characteristics use and the adaptation will be evaluated by an local set of experts in this technology use.

INTRODUCCIÓN

El código desarrollado en base a la tecnología COM+ es cuantioso en el mercado peruano como en el internacional , con la llegada de los Servicios Web[1] este código puede ser expuesto y utilizado por diferentes plataformas , existe un inconveniente al intentar exponer las transacciones de los componentes COM+ en actividades transaccionales multiplataforma todo o nada[2], la tecnología transaccional COM+ no fue desarrollada con estándares abiertos que permitan su participación en actividades transaccionales con otras plataformas.

Existe un trabajo conjunto de las empresas BEA Systems, International Business Machines Corporation y Microsoft - 2005, en la definición de un estándar de coordinación de actividades transaccionales[3], este estándar define estructuras de datos y procesos de comunicación entre diferentes elementos de software multiplataforma, para que ellos puedan participar en una actividad transaccional todo o nada, como menciona Clements Vasters (2005)[4], en su discusión respecto al referido estándar , existen varias características del estándar que se relacionan con los conceptos y bloques de construcción de la tecnología transaccional COM+, pero quedan varios conceptos que no tienen equivalencia, y no queda claro como el estándar pueda ser implementado utilizando la tecnología COM+.

---

[1] El diseño de la presente propuesta fue basada en los conceptos de las referencias bibliográficas 4,5,6 y 7.

[2] Referirse al Glosario para el término actividades transaccionales multiplataforma todo o nada

[3] Revisar el ANEXO A - DISEÑO DE LA SOLUCIÓN, sección EXPOSICIÓN DE COM+ EN ACTIVIDADES TRANSACCIONALES MEDIANTE SERVICIOS WEB para mayor detalle

[4] Para mayor detalle revisar el URL http://radio.weblogs.com/0108971/2002/08/16.html

Es por ello que el presente trabajo sienta la base para exponer las transacciones COM+ en actividades multiplataforma, de esta manera el código escrito actualmente en componentes COM+ podrá ser utilizado en actividades transaccionales multiplataforma, sin necesidad de ser reescrito, con lo que se daría soporte a una de las bases fundamentales del desarrollo de software, la reutilización de código, así se podrán ahorrar tiempo y dinero en el desarrollo de nuevos sistemas.

CAPITULO I - VALIDACIÓN DE LA HIPOTESIS

1.1. Introducción.

En el presente capítulo se detalla la validación de la hipótesis. La validación de la hipótesis se realizará en dos niveles, el primero será un prototipo que hace uso de un servicio Web que tiene implementado el diseño transaccional propuesto, seguiremos el flujo de la figura Número uno, para probar que el diseño se comporta de la forma planteada. El segundo será la validación de la hipótesis mediante el análisis de resultados de una encuesta realizada a especialistas en el manejo de la plataforma .NET y la tecnología transaccional COM+, de esta manera se evaluará la adecuación de la tecnología COM+ Transaccional al diseño propuesto en el presente trabajo.

1.2. Demostración con Prototipo

En esta primera parte de demostraremos y evaluaremos a través de un prototipo la participación del modelo COM+ en una actividad transaccional todo o nada, expuesta a través de un servicio Web, como se muestra en la Figura 1 y se detalla en su narración:

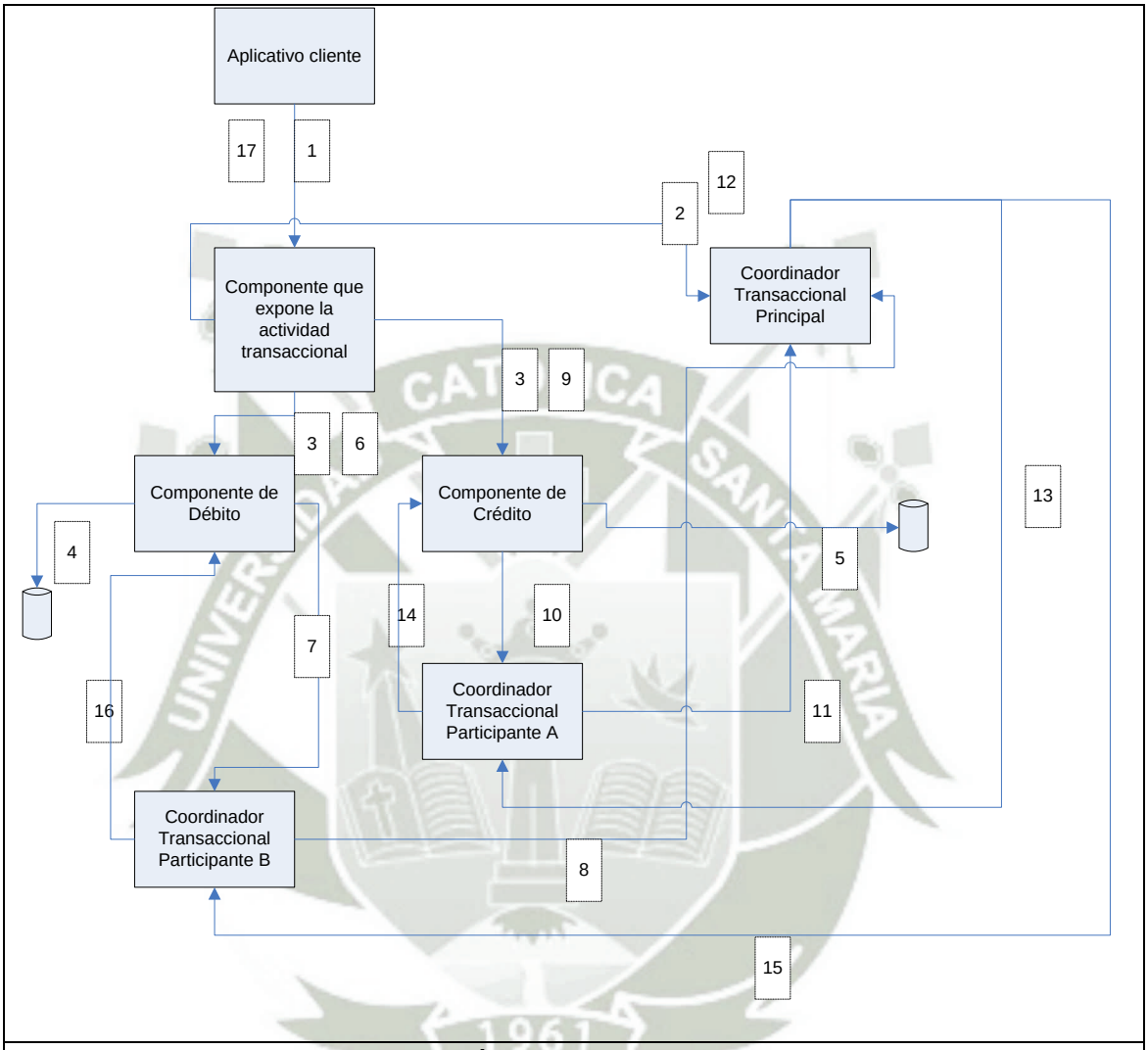


Figura 1. Proceso de demostración de comportamiento de Prototipo.

Fuente Elaboración Propia.

A. Narración:

- Mensaje 1: Por medio de este mensaje un aplicativo cliente solicitará la ejecución de una actividad transaccional “Transferencias Bancarias”.

- Mensaje 2: El componente de actividad transaccional le pide al Coordinador de transacciones principal que genere un contexto transaccional principal, el cual será utilizado en las llamadas transaccionales a otros componentes.
- Mensaje 3: El componente de actividad transaccional instancia y llama a los métodos de componente de débito y de crédito respectivamente, estos componentes instancias sus respectivos componentes COM+ y ejecutan sus respectivos códigos.
- Mensaje 4 y 5: Los componentes COM+ ejecutan sus actividades transaccionales sobre sus respetivas bases de datos.
- Mensaje 6: El componente de actividad transaccional llama a la interfaz transaccional del Componente débito pasándole el contexto transaccional principal.
- Mensaje 7: El componente de débito llama a su coordinador transaccional participante y le pide que cree un contexto transaccional participante, luego le pide a este coordinador que se comunique con el coordinador transaccional principal, para informar de su participación en la actividad transaccional.
- Mensaje 8: El coordinador transaccional participante se matrícula en la actividad transaccional, registrándose con el coordinado transaccional principal.
- Mensaje 9: El componente de actividad transaccional llama a la interfaz transaccional del Componente débito pasándole el contexto transaccional principal.

- Mensaje 10: El componente de crédito llama a su coordinador transaccional participante y le pide que cree un contexto transaccional participante, luego le pide a este coordinador que se comunique con el coordinador transaccional principal, para informar de su participación en la actividad transaccional.
- Mensaje 11: El coordinador transaccional participante se matrícula en la actividad transaccional, registrándose con el coordinado transaccional principal.
- Mensaje 12: El componente de actividad transaccional llama al coordinador principal, para que le indique cual es el resultado de su actividad transaccional.
- Mensaje 13: El coordinador transaccional principal llama al coordinador transaccional participante, para participar en el proceso que determina el resultado de la actividad transaccional, el participante debe informar sobre el resultado de la actividad transaccional.
- Mensaje 14: El coordinador transaccional participante llama al componente de crédito, para que participar en el proceso que determina el resultado de la actividad transaccional.
- Mensaje 15: El coordinador transaccional principal llama al coordinador transaccional participante, para participar en el proceso que determina el resultado de la actividad transaccional, el participante debe informar sobre el resultado de la actividad transaccional.

- Mensaje 16: El coordinador transaccional participante llama al componente de débito, para que para participar en el proceso que determina el resultado de la actividad transaccional.
- Mensaje 17: Se le entrega una respuesta al aplicativo cliente.

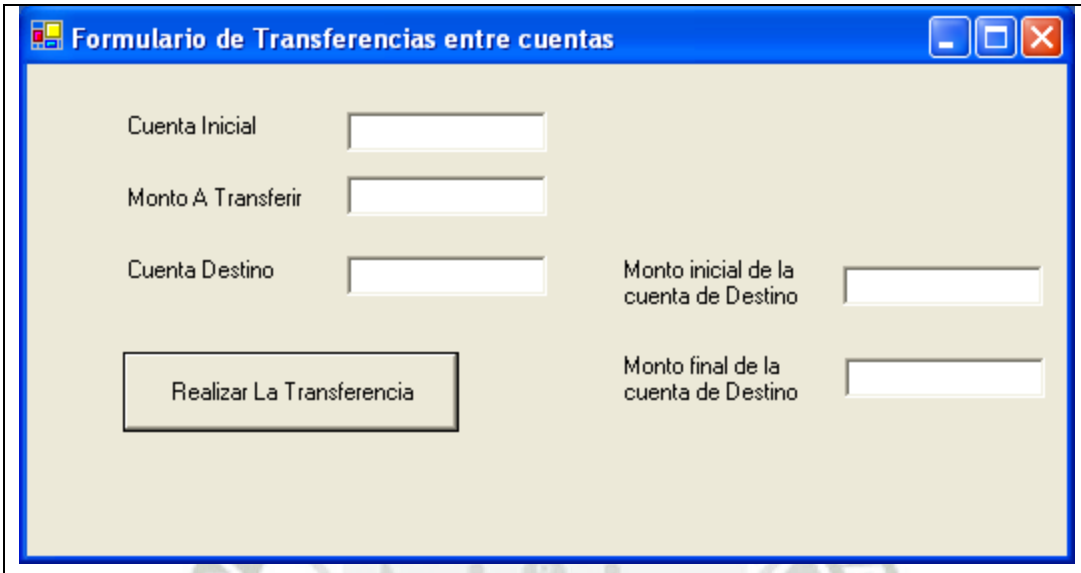


Figura 2. Pantalla inicial de cliente.

Fuente: Elaboración Propia.

Desde la pantalla inicial de cliente, podremos lanzar una transacción de transferencia entre dos cuentas diferentes, esta transacción iniciará el mensaje uno (ver la figura 1).

B. Comentarios y Discusiones :

En la opinión del autor: Queda demostrada, mediante la ejecución del prototipo que se pueden exponer componentes COM+ en actividades transaccionales basadas en servicios Web, adicionalmente se probó la funcionalidad del diseño propuesto en el presente trabajo.

Adicionalmente se hace uso del patrón orquestador[5] de servicios para llamar tanto al componente de débito como el de crédito, esta es una muy buena práctica para componer servicios que se encuentren en múltiples plataformas. Al utilizar el patrón orquestador y otros patrones de diseño de software se aseguran distintas características de ingeniería de software, tales como la calidad, estabilidad, desacoplamiento[6], es por esta razón unida al empleo de casos de uso que delimitan el ámbito de uso de aplicativo, que solo se necesita un prototipo para probar la aplicabilidad del diseño .

Queda demostrado que las modificaciones al estándar WS-COORDINATION, así como el uso de características adecuadas de del modelo transaccional COM+ funciona, mediante la implementación del diseño propuesto.

1.3. Demostración por medio de encuestas

El objetivo de la encuesta es determinar si la utilización de las características transaccionales del modelo COM+ son adecuadas para la propuesta realizada, se encuestarán a las personas que trabajan con la tecnología COM+.

Se elegirá como universo de investigación a los profesionales que tengan los conocimientos certificados por Microsoft, en la utilización profesional de la tecnología COM+, estos profesionales deben poseer la certificación MCAD Microsoft Application Developer , la cual como uno

[5] Referencia bibliográfica 1
[6] Referencia bibliográfica 2

de sus componentes incluye la programación de los componentes COM+, en la ciudad de Arequipa Perú hasta el mes de diciembre del año 2006 se tiene la siguiente relación de profesionales de sistemas certificados: 11 certificados Oficiales. [7].

A estos certificados se les aplicó la siguiente encuesta, previa exposición de las actividades que realiza la infraestructura Transaccional, el protocolo durable de dos fases y finalmente la exposición de los componentes COM+ en base al prototipo desarrollado en la demostración uno.

Se toma el universo en su totalidad.

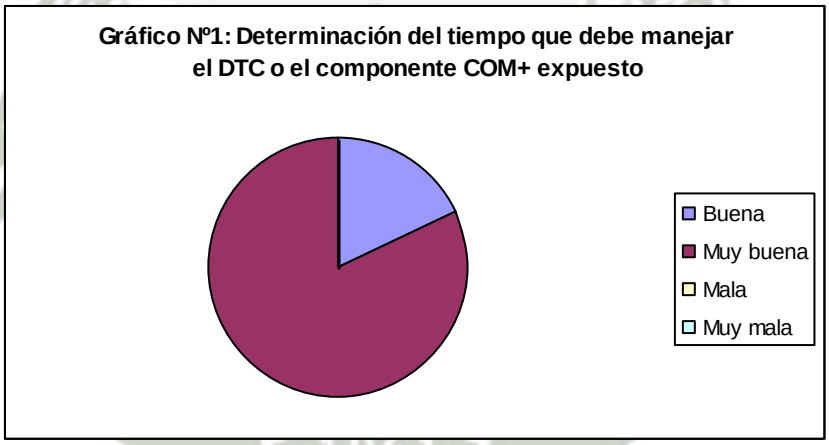
1.3.1. Resultado de la encuesta

Los resultados de la encuesta tomada se presentan en los siguientes cuadros:

1.3.1.1. Determinación del tiempo que debe manejar el DTC[8] o el componente COM+ expuesto

[7] Este dato fue obtenido a través de la institución New Horizons Perú
[8] Para mayor referencia sobre el DTC referirse al glosario.

Cuadro N°1: Determinación del tiempo que debe manejar el DTC o el componente COM+ expuesto. **Pregunta : ¿Cómo considera la determinación del tiempo que debe manejar el DTC o el componente COM+ expuesto?** **Fuente: Elaboración propia.**	
Buena	2
Muy buena	9
Mala	0
Muy mala	0



Fuente: Elaboración propia.

a. Interpretación del Resultado:

El 89% de los encuestados considera que el tiempo que maneja el DTC o el componente COM+ expuesto es muy Buena, el 18% considera que es buena, y ningún encuestado considero que fuera mala o muy mala.

b. Comentarios:

"Se podría buscar una métrica, respecto al tiempo máximo para ser utilizado en la transacción COM+, utilizando tipos de aplicativos y estadísticas acerca de las redes utilizadas." Jean Pierre Valencia[9]

Para lograr la adecuación del modelo COM+, que sea más adecuado varios tipo de aplicativos, se puede buscar está métrica en un futuro trabajo de investigación, es opinión del autor que esto dependerá mucho del tipo de aplicativos que corran en una determinada máquina, un diseño para un aplicativo determinado puede indicar un tiempo máximo de una transacción, este tiempo puede no ser aceptable en otro tipo de aplicativo.

1.3.1.2. Efecto la desactivación automática del componente expuesto luego de la llamada al método transaccional

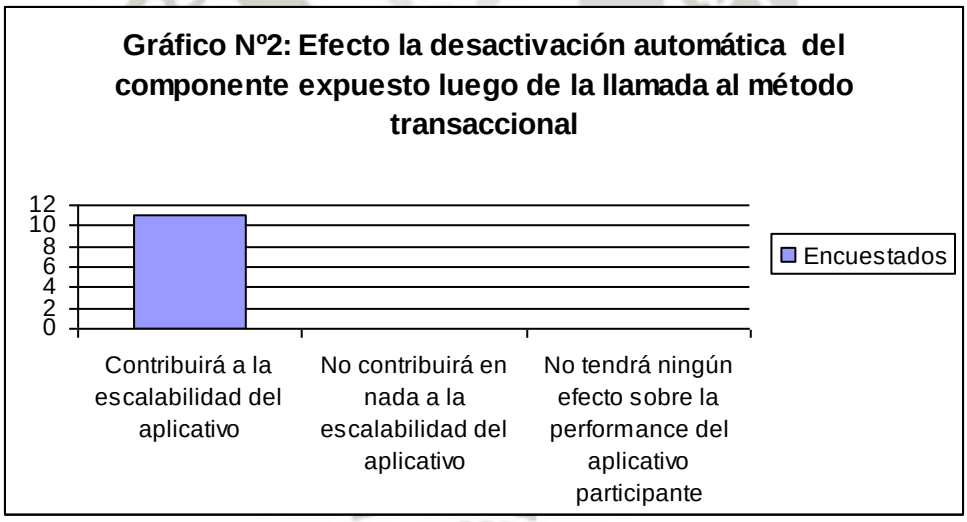
[9] Jean Pierre Valencia es uno de los participantes de la encuesta, y el comentario en referencia es producto de su respuesta a la pregunta 6 de más misma referida a los comentarios y acotaciones, para mas detalle ver el formato de la encuesta en el punto 3.1. Encuesta propuesta del Capitulo I – Validación de la Hipótesis del presente trabajo de investigación.

Cuadro N°2: Efecto la desactivación automática del componente expuesto luego de la llamada al método transaccional.

Pregunta: ¿Cómo considera que afectará la desactivación automática del componente expuesto luego de la llamada al método transaccional, no importando el resultado del mismo?

Fuente: Elaboración propia

Contribuirá a la escalabilidad del aplicativo	11
No contribuirá en nada a la escalabilidad del aplicativo	0
No tendrá ningún efecto sobre la performance del aplicativo participante	0



Fuente: Elaboración propia

a. Interpretación del Resultado:

El 100% de los encuestados considera que la desactivación automática del componente expuesto luego de la llamada al método transaccional contribuirá a la escalabilidad del aplicativo, este resultado descarta totalmente que dicha desactivación no contribuya

en nada a la escalabilidad del aplicativo o que no tenga ningún efecto sobre la performance del aplicativo participante.

b. Comentarios:

En la opinión del autor , para lograr la adecuación del modelo COM+, en la exposición hacia las actividades transaccionales multiplataforma , se puede lograr un amento en la carga que maneja un determinado componente si este trabaja sin estado , esto se logra utilizando la característica JIT de los componentes COM+ , de tal manera que luego de una llama a un método el componente se desactiva , inicializando todos sus atributos internos y quedando en memoria preparado para dar respuesta ante una nueva llamada , así un componente puede manejar gran cantidad de peticiones sin tener que recrearse por completo para cada petición.

1.3.1.3. Implementación de la clase de fachada y su respectivo almacenamiento en las variables de aplicación a nivel del aplicativo participante

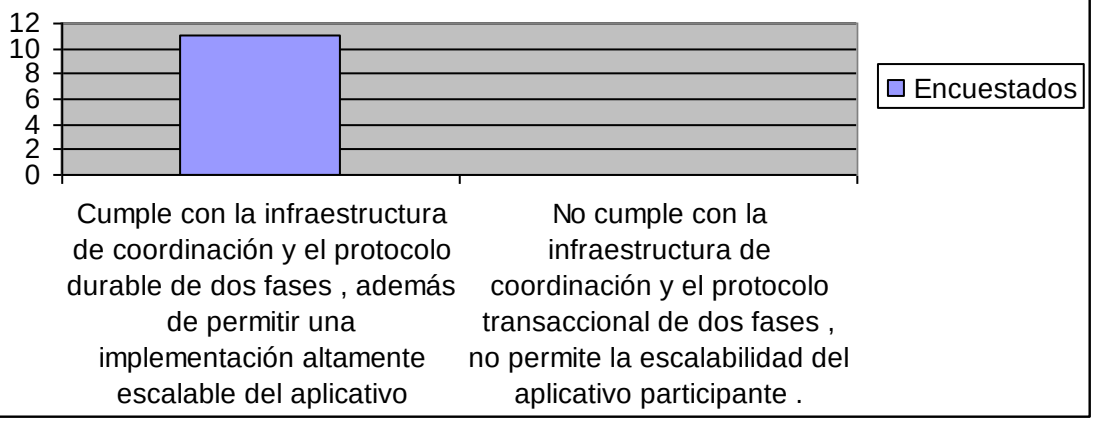
Cuadro N°3: Implementación de la clase de fachada y su respectivo almacenamiento en las variables de aplicación a nivel del aplicativo participante.

Pregunta: ¿Qué tan adecuada considera la implementación de la clase de fachada y su respectivo almacenamiento en las variables de aplicación a nivel del aplicativo participante?

Fuente: Elaboración propia

Cumple con la infraestructura de coordinación y el protocolo durable de dos fases , además de permitir una implementación altamente escalable del aplicativo participante	11
No cumple con la infraestructura de coordinación y el protocolo transaccional de dos fases , no permite la escalabilidad del aplicativo participante .	0

Gráfico Nº3: Implementación de la clase de fachada y su respectivo almacenamiento en las variables de aplicación a nivel del aplicativo participante



Fuente: Elaboración propia

a. Interpretación del Resultado:

Ningún encuestado consideró que la implementación de la clase fachada y su respectivo almacenamiento en las variables de aplicación a nivel del aplicativo participante no cumple con la infraestructura de coordinación y el protocolo transaccional de dos fases, no permitiendo la escalabilidad del aplicativo participante; al contrario, los encuestados indicaron que si cumple con la infraestructura de coordinación y el protocolo durable de dos fases, además permite una implementación altamente escalable del aplicativo participante.

b. Comentarios:

"Es una implementación interesante de la especificación, es importante que los coordinadores almacenen de forma temporal la información de las transacciones en las que participan, pero habría que ver un mecanismo de acceso que incremente la velocidad con las que se administran estas variables". Paul Obando[10]

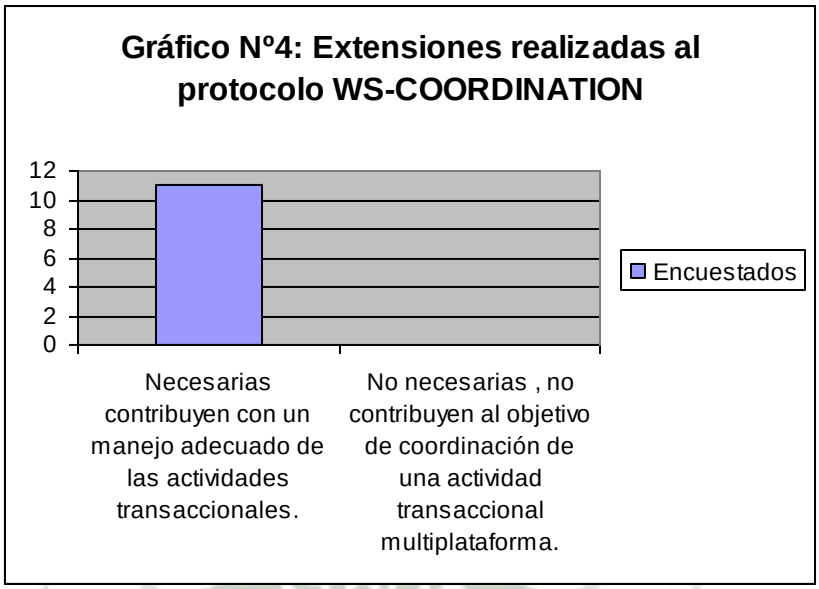
En la opinión del autor: Al utilizar la implementación de los coordinadores como servicios Web, se debe utilizar alguna forma de almacenamiento de estado para que los coordinadores recuerden la información de las actividades transaccionales en las que participan , en el prototipo

[10] Paúl Obando es uno de los participantes de la encuesta, y el comentario en referencia es producto de su respuesta a la pregunta 6 de más misma referida a los comentarios y acotaciones, para mas detalle ver el formato de la encuesta en el punto 3.1. Encuesta propuesta del Capitulo I – Validación de la Hipótesis del presente trabajo de investigación.

elaborado se utilizaron las variables almacenadas en variables de sesión, recayendo en los algoritmos de búsqueda que este objeto provee , se puede buscar en un futuro trabajo de investigación una forma más eficiente de administración de estos objetos , pudiendo utilizar árboles de búsqueda , caches intermedios de búsqueda , etc. Por el momento se puede ver el resultado de los tiempos que se consumen en la ejecución de los métodos utilizando el conjunto de clases Trace y Debug disponibles en el espacio de nombres System.Diagnostics.

1.3.1.4. Extensiones realizadas al protocolo WS-COORDINATION

Cuadro N°4: Extensiones realizadas al protocolo WS-COORDINATION. **Pregunta: ¿Cómo considera las extensiones realizadas al protocolo WS-COORDINATION?** **Fuente: Elaboración propia**	
Necesarias contribuyen con un manejo adecuado de las actividades transaccionales.	11
No necesarias , no contribuyen al objetivo de coordinación de una actividad transaccional multiplataforma.	0



Fuente: Elaboración propia

a. Interpretación del Resultado:

El 100% de los encuestados consideró que las extensiones realizadas al protocolo WS-CORDINATION son necesarias y contribuyen con un manejo adecuado de las actividades transaccionales.

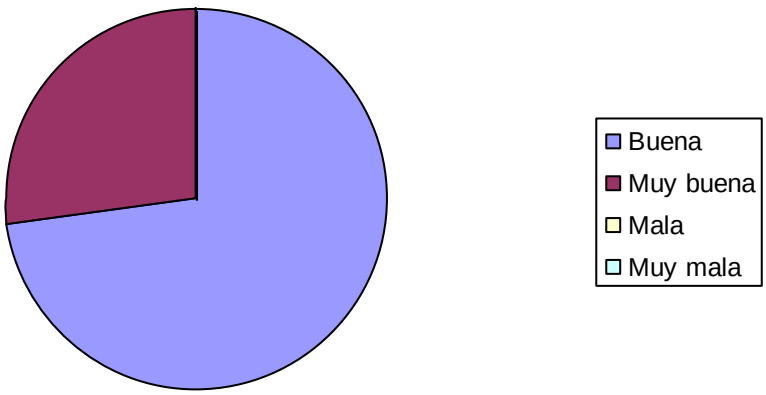
b. Comentarios :

En la opinión del autor las modificaciones a las estructuras de datos fueron necesarias, tales como las clases que representan al contexto transaccional y el tiempo en el cual se realizan las transacciones.

1.3.1.5. Exposición del Modelo Transaccional COM+ para su Utilización En Actividades Transaccionales Todo o Nada Basadas en Servicios Web

Cuadro N°5: Exposición del Modelo Transaccional COM+ para su Utilización En Actividades Transaccionales Todo o Nada Basadas en Servicios Web. **Pregunta: ¿Cómo considera en líneas generales la Exposición del Modelo Transaccional COM+ para su Utilización En Actividades Transaccionales Todo o Nada Basadas en Servicios Web ?.** **Fuente: Elaboración propia**	
Muy buena	3
Buena	8
Mala	0
Muy mala	0

Gráfico N°5: Exposición del Modelo Transaccional COM+ para su Utilización En Actividades Transaccionales Todo o Nada Basadas en Servicios Web



Fuente: Elaboración propia.

a. Interpretación del Resultado:

El 73% de los encuestados consideró que la exposición del modelo transaccional COM+ para su utilización en actividades transaccionales todo o Nada basadas en servicios WEB es buena, mientras que el 27% consideró que era muy buena. Cabe resaltar que ningún encuestado considera que dicha exposición es mala o muy mala.

b. Comentarios :

- "Las organizaciones que utilicen que utilicen esta propuesta deberían implementarla teniendo en cuenta un bajo acoplamiento entre los

elementos participantes de la actividad transaccional." Jean Pierre Valencia[11].

En la opinión del autor esta debe ser una característica fundamental de la implementación completa del presente trabajo, se puede garantizar que los coordinadores transaccionales puedan ser utilizados por otros componentes, de otras plataformas para realizar su trabajo transaccional.

- "Se debe tener en cuenta el movimiento de los trabajos en esta rama, en especial los que administra el W3C, para alinear a esos estándares la presente investigación." Paul Obando[12].

  En la opinión del autor, al utilizar estas consideraciones se desarrollará un trabajo que pueda interactuar con el software y marcos de trabajo que se implementen en un futuro.

[11] Jean Pierre Valencia es uno de los participantes de la encuesta, y el comentario en referencia es producto de su respuesta a la pregunta 6 de más misma referida a los comentarios y acotaciones, para mas detalle ver el formato de la encuesta en el punto 3.1. Encuesta propuesta del Capitulo I – Validación de la Hipótesis del presente trabajo de investigación.

[12] Paúl Obando es uno de los participantes de la encuesta, y el comentario en referencia es producto de su respuesta a la pregunta 6 de más misma referida a los comentarios y acotaciones, para mas detalle ver el formato de la encuesta en el punto 3.1. Encuesta propuesta del Capitulo I – Validación de la Hipótesis del presente trabajo de investigación.

CONCLUSIONES

1. Fue posible diseñar una forma adecuada en la cual los componentes desarrollados bajo la tecnología COM+ puedan participar en actividades transaccionales todo o nada entre múltiples plataformas por medio del uso de un estándar abierto como es el XML.

2. Se demostró que la exposición de los componentes COM+ hacia actividades transaccionales multiplataforma puede hacerse de forma adecuada sin entrar en conflicto con los atributos y características de los servicios transaccionales de la tecnología COM+. Esta demostración tuvo lugar al elaborar un prototipo y se corroboró la adecuación del modelo transaccional COM+ con la realización de las encuestas cuyos resultados, en general, indicaron que la determinación del tiempo que debe manejar el DTC o el componente COM+ expuesto es en su mayoría muy bueno o bueno, y que además afecta la desactivación automática del componente expuesto luego de la llamada al método transaccional; contribuyendo a la escalabilidad del aplicativo y la implementación de la clase fachada y su respectivo almacenamiento en las variables de aplicación a nivel del aplicativo participante cumple con la infraestructura de coordinación y que el protocolo durable de dos fases, además de permitir una implementación altamente escalable del aplicativo participante.

3. Los estándares WS-COORDINATION y WS-TRANSACTION no especifican actualmente todos los detalles de manejo de actividades transaccionales multiplataforma, estos estándares fueron ampliados en el presente trabajo para lograr un diseño preparado para ser implementado.(Ver anexo C y D)

4. El acercamiento más estable desde el punto de vista de los aplicativos participantes es la Exposición ACID[13] completa al protocolo durable de dos fases, este tipo de exposición cumple con todas las propiedades requeridas para el manejo de recursos transaccionales. (Ver anexo A, EL PROTOCOLO DURABLE DE DOS FASES – Casos de Uso).

5. Se pudieron adaptar patrones de diseño[14], tales como: El patrón de Fachada, Experto, bajo acoplamiento, alta Cohesión, lo cual garantiza la adecuación, calidad de la propuesta. (Ver anexo A casos de uso y pseudocódigo INFRAESTRUCTURA DE COORDINACIÓN y EL PROTOCOLO DURABLE DE DOS FASES).

6. El dominio de la aplicabilidad del modelo es exponer componentes COM+ en actividades transaccionales todo o nada, estos componentes pueden ser componentes existentes o componentes que se creen desde cero.

[13] El término ACID se encuentra definido en el glosario del presente trabajo.
[14] El término patrones de diseño se encuentra definido en el glosario del presente trabajo.

7. La aplicabilidad del presente trabajo es adecuada en aplicaciones bancarias y flujos transaccionales donde el tiempo de espera no sea prohibitivo, la limitación del uso del presente trabajo la constituye el tiempo de respuesta que dura en terminar una transacción multiplataforma, este tiempo dependerá de factores tales como la latencia de red, tiempo de interpretación del estándar XML, llamadas a los componentes locales, un ejemplo de las aplicaciones que tienen estos tiempos prohibitivos son: Control en tiempo real de componentes de hardware, motores, calderas. Etc.Por lo

RECOMENDACIONES

1. Esta propuesta puede ser completada en la capa de exposición de las actividades transaccionales y en ese punto es adaptable a los estándares que publique el World Wide Web Consortium W3C, para mantener alineación en los estándares utilizados por la industria.

2. Para reutilizar esta propuesta en diferentes plataformas, la implementación se debe realizar bajo la plataforma .NET en lo que se refiere al Aplicativo Participante, puesto a que por el momento la plataforma Windows ofrece un mayor control sobre el servidor COM+, que es el elemento que permite ejecutar los componentes COM+, los demás elementos participantes en un actividad transaccional pueden implementarse en cualquier otra plataforma de desarrollo disponible en el mercado.

3. La exposición de otros estándares de manejo de componentes transaccionales como lo son CORBA[15] y RMI[16] pueden seguir este mismo marco de exposición, a través de clases de fachada que se encarguen de coordinar internamente las características específicas de estos estándares, en los aplicativos participantes.

[15] El término CORBA se encuentra definido en el glosario del presente trabajo.

[16] El término RMI se encuentra definido en el glosario del presente trabajo.

4. La comunicación de todos los elementos que participan en una actividad transaccional debería realizarse a través de mensajes SOAP sin el uso Proxies , para evitar las ligaduras dependiente entre elementos.

BIBLIOGRAFÍA

1. Libros

1 Fowler , Martin (2003). “Patterns of Enterprise Application Architecture ” . Pearson Educations .

2 Craig Larman (1999). "Uml y Patrones". Prentice Hall Hispanoamerica S.A. - México .

3 Jacobson, Ivar; Booch, Grady; Rumbaugh, James. (2000). "El Lenguaje Unificado de Modelado" Editorial Addison Wesley. Madrid

4 Microsoft Corporation (2002). "Developing XML Web Services Using Microsoft ASP.NET - Workbook". MSDN

5 Microsoft Corporation (2002). "Building COM+ Applications Using Microsoft .NET Enterprise Services - Workbook". MSDN

6 Roger S. Pressman (2002). “Ingeniería del Software un Enfoque Práctico – Quinta Edición ”. Editorial Mc Graw Hill .

7 Ricardo Eíto Brun(2001) . “Programación con XML”. Editorial Anaya Multimedia – Madrid.

2. Direcciones URL

- Propiedades ACID
  http://www.microsoft.com/latam/technet/articulos/200005/art22/
- WS-AtomicTransaction
  http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-AtomicTransaction.pdf

- Ws-coordination

  http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-Coordination.pdf

ANEXOS

ANEXO A - PROYECTO DE INVESTIGACIÓN

Universidad Católica de Santa María

Escuela de Post Grado

Maestría en Ingeniería de Software

Exposición del Modelo Transaccional COM+ para su Utilización en Actividades Transaccionales Todo o Nada Basadas en Servicios Web

Proyecto presentado por el Bachiller:

Hugo Christian Montenegro Beltrán

Para optar el Grado Académico de

Magíster en Ingeniería de Software

Arequipa – enero de 2006

PLANTEAMIENTO TEÓRICO

1. Preámbulo

En el desarrollo de mi trabajo actual, en la institución Banco de Crédito del Perú, hay gran cantidad de componentes de software desarrollados en diferente plataformas, una de esas plataformas es la plataforma Windows en la cual existen componentes desarrollados bajo el modelo COM+[17], con la aplicación de SOA[18] el Banco intenta crear servicios comunes con funcionalidad reutilizable en diferentes aplicativos , varias funcionalidades resultan desarrolladas en base a la unión en actividades transaccionales de varios servicios comunes, entre estos servicios comunes se tienen servicios implementados bajo el modelo COM+ , estos no pueden trabajar en actividades transaccionales multiplataforma todo o nada, en vez de ello se tiene que modificar su código para aceptar métodos de compensación , es decir , métodos que intenten revertir el trabajo realizado por una llamada previa al componente , no todos las actividades tienen una funcionalidad adecuada utilizando este acercamiento , por lo que es necesario diseñar e implementar una forma en la que los componentes COM+ trabajen en una actividad transaccional multiplataforma todo o nada , evitando en lo posible el tener que modificar el código de los componentes existentes , es por ello el esfuerzo de el presente trabajo en intentar demostrar que se puede

[17] Ver el Anexo B – Glosario para más detalle de COM+

[18] Ver el Anexo B – Glosario para más detalle de SOA (Service-Oriented Architecture)

exponer el Modelo Transaccional COM+ para su Utilización en Actividades Transaccionales Todo o Nada Basadas en Servicios Web, mediante Servicios Web[19], porque cumplen con un estándar abierto y accesible desde múltiples plataformas .

[19] Para mayor detalle referente a servicios Web revisar el punto 3.4.Servicios Web del Marco Conceptual del Anexo A - PROYECTO DE INVESTIGACIÓN del Presente trabajo de investigación.

2. El Problema de investigación

2.1. Enunciado del Problema

"Exposición del Modelo Transaccional COM+ Para su Utilización en Actividades Transaccionales Todo o Nada Basadas en Servicios Web".

2.2. Descripción del Problema

Las transacciones todo o nada son muy importantes en la actualidad porque permiten integrar en un proceso , el trabajo de varios componentes de software implementados en diferentes plataforma , tales como Windows , Os 390 , Linux , Aix , etc. El software desarrollado actualmente bajo el modelo COM+ no puede ser invocado desde cualquier plataforma, pero sus transacciones no pueden participar en actividades transaccionales multiplataforma, es por ello el presente trabajo busca realizar la exposición de las transacciones implementadas en el modelo COM+ a este tipo de actividades, y así colaborar con la reutilización de código implementado en componentes COM+ con el ahorro de tiempo y dinero que implica este hecho.

A continuación se detalla el Tipo, Nivel y Área de Investigación del presente trabajo:

a) Tipo de Investigación **:** Descriptiva.

b) Nivel de Investigación **:** Relacional.

c) Área de la investigación **:** Ciencias Físicas y formales – Ingeniería de Software.

La investigación tiene características del tipo aplicada porque vamos a ver la utilidad de los patrones existentes y/o creados en el desarrollo de servicios Web. Posee además características del tipo Fundamental por la naturaleza de los patrones, buscar un principio o generalización para el diseño de software.

El nivel de Investigación es Relacional, porque vamos a analizar la relación entre dos variables significativas de esta investigación: los patrones propuestos y el comportamiento de los sistemas creados a partir de estos patrones.

2.3. Justificación

Actualmente se pueden desarrollar aplicativos altamente funcionales, escalables, robustos, a partir de la conjunción de funcionalidades que pueden proveer varios Servicios Web.

Cada servicio Web utilizado provee métodos que pueden ser llamados por el aplicativo que desea utilizarlos, el servicio Web provee un manejo transaccional interno al método que se invoca, de tal manera que por ejemplo, una operación de transferencia entre dos cuentas bancarias de diferentes bancos implementadas por dos servicios Web distintos, desencadenadas por un aplicativo que llame a ambas, no pueden

compartir la misma actividad transaccional todo o nada como las que exponen los protocolos transaccionales de dos fases.

El manejo de toda la actividad transaccional de esta forma puede dejar la información en estado no consistente, por ejemplo si se puede debitar la primera cuenta en el primer banco y si la operación de abono en el segundo banco falla, la suma de dinero en ambas cuantas quedaría alterado, respecto a este mismo valor antes de realizar la operación.

El área de software a la que pertenecen los servicios Web es relativamente joven en el mercado , necesita nutrirse de sólidos patrones de diseño y estándares de desarrollo de software para conseguir entre otras cosas la participación de los servicios Web en actividades transaccionales conjuntas a otros servicios Web o Aplicativos Participantes . Actualmente en el mercado mundial existe gran cantidad de código desarrollado e implementado bajo el modelo de objeto componente COM+, se tendría un ahorro en costos de mantenimiento de los sistemas y desarrollo de nuevos si se tuviera un marco de trabajo que permita exponer esta funcionalidad a través de actividades transaccionales todo o nada basadas en servicios Web.

3. Marco Conceptual

3.1. COM+ Component Object Model – Modelo de objeto componente

Según Wikipedia 2006[20] Component Object Model (COM) es una plataforma de Microsoft para componentes de software introducida por dicha empresa en 1993. Esta plataforma es utilizada para permitir la comunicación entre procesos y la creación dinámica de objetos, en cualquier lenguaje de programación que soporte dicha tecnología. El término COM es a menudo usado en el mundo del desarrollo de software como un término que abarca las tecnologías OLE, OLE Automation, ActiveX, COM+ y DCOM. Si bien COM fue introducido en 1993, Microsoft no hizo énfasis en el nombre COM hasta 1997.

Aunque estas tecnologías han sido implementadas en muchas plataformas, son principalmente usadas con Microsoft Windows. Se espera que COM sea sustituido, al menos en un cierto grado, por Microsoft .NET.

3.2. DCOM Distributed Component Object Model (DCOM)[21]

En español Modelo de Objetos de Componentes Distribuidos, Según Wikipedia 2006es una tecnología propietaria de Microsoft para desarrollar componentes software distribuidos sobre varios ordenadores y que se comunican entre sí. Extiende el modelo COM de Microsoft y proporciona el sustrato de comunicación entre la infraestructura del

[20] Revisar la URL http://es.wikipedia.org/wiki/Component_Object_Model para mayor información.

[21] IDEM 10.

servidor de aplicaciones COM+ de Microsoft. Ha sido abandonada en favor del framework .NET

La adición de la "D" a COM fue debido al uso extensivo de DCE/RPC, o más específicamente la versión mejorada de Microsoft, conocida como MSRPC.

En términos de las extensiones que añade a COM, DCOM tenía que resolver los problemas de

Aplanamiento - Serializar y deserializarlos argumentos y valores de retorno de las llamadas a los métodos "sobre el cable".
Recolección de basura distribuida, asegurándose que las referencias mantenidas por clientes de las interfaces sean liberadas cuando, por ejemplo, el proceso cliente ha caído o la conexión de red se pierde.
Uno de los factores clave para resolver estos problemas es el uso de DCE/RPC como el mecanismo RPC subyacente bajo DCOM. DCE/RPC define reglas estrictas en cuanto al aplanamiento y a quién es responsable de liberar la memoria.

DCOM fue uno de los mayores competidores de CORBA. Los defensores de ambas tecnologías sostenían que algún día serían el modelo de código y servicios sobre Internet. Sin embargo, las dificultades que suponía conseguir que estas tecnologías funcionasen a

través de cortafuegos y sobre máquinas inseguras o desconocidas, significó que las peticiones HTTP normales, combinadas con los navegadores web les ganasen la partida. Microsoft, en su momento intentó y fracasó anticiparse a esto añadiendo un transporte extra HTTP a DCE/RPCdenominado "ncacn_http" (Connection-based, over HTTP).

3.3. Xml

Según Wikipedia 2006[22] , XML, sigla en inglés de eXtensible Markup Language («lenguaje de marcas extensible»), 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 (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

[22] Para mayor detalle remitirse a http://es.wikipedia.org/wiki/XML.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

3.4. Servicios Web

Según Wikipedia 2006[23] , Un servicio Web (en inglés Web service) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes y ejecutadas sobre cualquier plataforma pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

[23] Para poder revisar el articulo completo ver el URL http://es.wikipedia.org/wiki/Servicio_Web.

Estándares empleados:

- Web Services Protocol Stack: Así se denomina al conjunto de servicios y protocolos de los servicios Web.
- XML: Es el formato estándar para los datos que se vayan a intercambiar.
- SOAP o XML-RPC: Protocolos sobre los que se establece el intercambio.
- Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP, FTP, o SMTP.
- WSDL: Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.
- UDDI: Protocolo para publicar la información de los servicios Web. Permite a las aplicaciones comprobar qué servicios web están disponibles.
- WS-Security: Protocolo de seguridad aceptado como estándar por OASIS. Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados.

Ventajas de los servicios Web

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.

- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar.

Inconvenientes de los servicios Web

Para realizar transacciones no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA.

Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como RMI, CORBA, o DCOM. Es uno de los inconvenientes derivados de adoptar un formato basado en texto. Y es que entre los objetivos de XML no se encuentra la concisión ni la eficacia de procesamiento.

Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

Existe poca información de servicios web para algunos lenguajes de programación.

Razones para crear servicios Web

La principal razón para usar servicios Web es que se basan en HTTP sobre TCP en el puerto 80. Dado que las organizaciones protegen sus redes mediante firewalls -que filtran y bloquean gran parte del tráfico de Internet-, cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores. Los servicios Web se vehiculan por este puerto, por la simple razón de que no resultan bloqueados.

Otra razón es que, antes de que existiera SOAP, no había buenas interfaces para acceder a las funcionalidades de otros ordenadores en red. Las que había eran ad hoc y poco conocidas, tales como EDI, RPC, u otras APIs.

Una tercera razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más acusada.

3.5. Soap

Según Wikipedia 2006[24] , SOAP (siglas de Simple Object Access Protocol) es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

Tecnología

A diferencia de DCOM y CORBA, que son binarios, SOAP usa el código fuente en XML. Esto es una ventaja ya que facilita su lectura por parte de humanos, pero también es un inconveniente dado que los mensajes resultantes son más largos. El intercambio de mensajes se realiza mediante tecnología de componentes (software componentry, ver ingeniería de software). El término Object en el nombre significa que se adhiere al paradigma de la programación orientada a objetos.

SOAP es un marco extensible y descentralizado que permite trabajar sobre múltiples pilas de protocolos de redes informáticas. Los procedimientos de llamadas remotas pueden ser modelados en la forma de varios mensajes SOAP interactuando entre sí.

[24] Para mayor detalle visitar http://es.wikipedia.org/wiki/SOAP.

SOAP funciona sobre cualquier protocolo de Internet, generalmente HTTP, que es el único homologado por el W3C. SOAP tiene como base XML, con un diseño que cumple el patrón Cabecera-Desarrollo de diseño de software, como otros muchos diseños, verbigracia HTML. La cabecera Header es opcional y contiene metadatos sobre enrutamiento (routing), seguridad o transacciones. El desarrollo Body contiene la información principal, que se conoce como carga útil (payload). La carga útil se acoge a un XML Schema propio.

3.6. WS-Coordination[25]

Según Wikipedia 2006[26], es una especificación desarrollada por BEA Systems , IBM y Microsoft , que describe un marco de trabajo extensible para proveer protocolos que coordinenen las acciones de aplicaciones distribuidas . Estos protocolos de coordinación son utilizados para soportar un número de aplicaciones que incluye aquellas que necesitan llegar a un acuerdo consistente acerca del resultado de una actividad transaccional.

El marco de trabajo define en su especificación, la habilitación de un servicio de aplicación, para que este cree un contexto necesario para propagar una actividad transaccional hacia otros servicios y para registrar los protocolos de coordinación . El marco de trabajo constituye la base para habilitar el procesamiento de transacciones, flujos de

[25] Revisar el Anexo F - Web Services Coordination (Ws-Coordination) del presente trabajo de investigación para mayor información.

[26] Puede visitar el URL http://en.wikipedia.org/wiki/WS-Coordination para mayor detalle.

trabajo y otros sistemas existentes, hacia la coordinación y operación de ambientes heterogéneos, ocultando sus protocolos propietarios.

3.7. El estado de la ciencia en el diseño de los servicios Web XML

En este primer capítulo veremos parte del desarrollo de la tecnología de los Servicios Web XML, haciendo hincapié en aquellos temas en los cuales el presente trabajo de investigación intentará brindar soluciones o aportes.

3.8. Composición de los mensajes (Esquemas) y representación de Data

Como parte del contrato de comunicación entre los diferentes elementos que participarán en las actividades transaccionales, se debe dar significado al flujo que pasa por el canal de comunicación, es comparable con el lenguaje hablado en una línea telefónica, este significado consta de 3 partes:

- El formato de representación de la data.
- El esquema del Mensaje
- La data transmitida

3.8.1. El formato de representación de la data

“Si dos aplicaciones se pueden comunicar satisfactoriamente, ellas deben de estar de acuerdo en un conjunto común de definiciones para

la data que es pasada sobre la conexión. Los protocolos de los servicios web establecen restricciones sobre la data que pueden transportar, esta debe estar en un formato textual, Pero ¿que representa este formato textual? ¿Representa objetos serializados, arrays de enteros o documentos Xml?

Hay básicamente dos caminos para proveer información acerca de que representa la data, una descripción externa o una descripción de si misma por parte de la data. Una descripción externa define el esquema de la data en alguna manera externa a la data. Los lenguajes de definición de interface utilizados por Rpc, Dcom, Corba son ejemplos de lenguajes diseñados para escribir descripciones externas. Esta descripción externa limita la interoperatibilidad, todas las computadoras envueltas en la colaboración deben acceder a la descripción externa.

La descripción de si misma por parte de la data incluye una descripción empotrada en la data. Utilizando este acercamiento se incrementa la interoperatibilidad porque la data puede ser analizada sin tener que consultar una descripción externa a la data[27].

Los servicios Web utilizan XML como el formato de representación de la data, XML provee las siguientes ventajas[28]:

[27] Revisar el glosario del documento de la referencia 21 para mas detalle.

- Está basado en texto y por lo tanto compatible con el canal de comunicación.
- Es un estándar de la industria con un amplio soporte por parte de la industria y el usuario.
- Se describe por si mismo
- Es interoperable.
- Los interpretadores de XML existen virtualmente para todas las plataformas, y muchas herramientas de desarrollo hacen fácil el desarrollo de esquemas de aplicación XML. Además el estándar puede ser comprendido por las personas que lo analizan[29].

3.8.2. El esquema del Mensaje

El mensaje es la unidad fundamental de comunicación en los servicios Web y por lo tanto es imperativo que todas las partes que colaboran tengan un preciso entendimiento del contenido de el mensaje.

El contrato de comunicación debe especificar todos los mensajes asociados con la respuesta de un servicio, así los contenidos por cada mensaje deberían ser especificados, especificando el tipo de datos de sus elementos y especificando restricciones asociadas con los tipos o entre ellos.

[28] Marchal, Benoît, "Soluciones XML Aplicadas"; Sams Publishing; Agosto 2000.
[29] Referencia Bibliográfica 7

SOAP[30] divide el mensaje entre dos secciones: encabezado opcional y cuerpo obligatorio. La cabecera contiene información asociada con la comunicación y la infraestructura de servicios. El cuerpo contiene el contenido orientado al negocio del mensaje, mientras que la cabecera contiene la metadata del mensaje[31].

[30] Ervin, "Análisis Web Services (WS) Segunda Parte (SOAP, WSDL y UDDI)", Agosto 2005, http://www.activalink.net/display/PersonalInfo/Informe+Web+Services+WS+Segunda+Parte+SOAP,+WSDL+y+UDDI.

[31] Revisar el glosario del documento de la referencia 21 para mas detalle.

4. Análisis de antecedentes investigativos

4.1. WS-Coordination[32]

a. Título : Web Services Coordination (WS-Coordination) 2005

b. Autores: Luis Felipe Cabrera, Microsoft George Copeland, Microsoft Max Feingold, Microsoft (Editor) Robert W Freund, Hitachi Tom Freund, IBM Jim Johnson, Microsoft Sean Joyce, IONA Chris Kaler, Microsoft Johannes Klein, Microsoft David Langworthy, Microsoft Mark Little, Arjuna Technologies Anthony Nadalin, IBM Eric Newcomer, IONA David Orchard, BEA Systems Ian Robinson, IBM John Shewchuk, Microsoft Tony Storey, IBM

c. Objetivo:

Diseñar un marco de Trabajo para permitir la propagación de contextos transaccionales en una actividad transaccional.

d. Resultados:

Tuvo como resulta un marco de trabajo que incluye procesos de comunicación, estados de objetos que participan en la comunicación, estructuras de datos generales para la comunicación

e. Resumen:

Esta especificación describe un marco de trabajo extensible para proveer protocolos que coordinen acciones de aplicaciones distribuidas. Muestra como los protocolos de coordinación son utilizados para soportar un número de aplicaciones, incluyendo las

[32] Para mayor información revisar el Anexo F - Web Services Coordination (Ws-Coordination) del presente trabajo de investigación.

que necesitan llegar a un acuerdo sobre el resultado de actividades distribuidas.

Según la opinión del autor este framework, no todos los conceptos necesarios para exponer una tecnología transaccional específica , además de no contar con un diseño de proceso exacto , en el presente trabajo se buscará ahondar en estos temas respecto a la exposición de la tecnología transaccional COM+.

4.2. ComBridge[33]

a. **Título :** ComBridge 1997

b. **Autors:** Empresa Octated **.**

c. **Objetivo:**

Exponer el modelo Com+ para ser consumido y utilizado en las plataformas Unix y Linux.

d. **Resultados:**

Se pueden llamar objetos creados bajo el modelo Com+ , pero no se pueden utilizar todas sus características , en especial las características transaccionales del modelo .

e. **Resumen:**

ComBridge Permite a clients UNIX y LINUX accesar a objetos COM , DCOM y COM+ que corren en la plataforma WIN32 . El soporte incluye el manejo de objetos ADO que se devuelven como resultado a la llamada a una función , tales como los Recordsets . ComBridge

[33] Para revisar el trabajo de investigación puede visitar el URL http://www.octatec.co.uk/CB_document.htm.

tiene embebido un wrapper para facilitar la comunicación CORBA/COM , y entre Windows y Unix .

El sistema es implementado como una librería de cliente y un servicio Win32 o programa de servidor para sistemas operativos Win32 que no soporten servicios.

En la opinión del autor, se pueden analizar las estructuras de datos utilizadas en la exposición del modelo COM+, pero estas resultan limitadas , puesto que dependen de las plataformas para las cuales el modelo COM+ es expuesto , en vez de utilizar un estándar comprendido por múltiples plataformas tal como el XML.

5. Objetivos

5.1. Objetivo General

Diseñar la forma en la cual se expondrá los componentes de software desarrollados bajo la tecnología COM+ para ser utilizados por medio de actividades transaccionales todo o nada basadas en Servicios Web.

5.2. Objetivos Secundarios

Independiente.

1. Complementar las propuestas sobre transacciones entre servicios Web existentes, tales como WS-COORDINATION y WS-TRANSACCTION[34] .

Dedependiente.

2. Adaptar los patrones de diseño de software existentes en la actualidad para utilizarlos en el desarrollo de Servicios Web.

[34] Para mayor información remitirse a los anexos C del presente trabajo de investigación.

6. Hipótesis

"Es posible diseñar una forma adecuada en la cual los componentes desarrollados bajo la tecnología COM+ puedan participar en actividades transaccionales todo o nada basadas en Servicios Web".

6.1. Variables

a. Independientes

- El diseño de la forma de exposición de las transacciones síncronas de los componentes COM+.

b. Dependientes

- La adecuación de la exposición respecto a la transaccionalidad de los componentes COM+.

6.2. Indicadores

3. Participación en el modelo transaccional de dos fases manejado por COM+ a través de un estándar de comunicación accesible desde múltiples plataformas

PLANTEAMIENTO OPERACIONAL

1. Técnicas , Instrumentos y Materiales de Verificación

Variables	Indicadores	Técnicas, Instrumentos y Materiales de Verificación
La adecuación de la exposición respecto al trabajo transaccional de los componentes COM+.	Conflicto entre las propiedades transaccionales que maneja COM + y las propiedades transaccionales utilizadas por la propuesta de la exposición.	Investigación y evaluación respecto al modelo transaccional que maneja COM+
	Opinión puntual acerca de la propuesta por parte de la comunidad de desarrolladores de la localidad	Técnicas estadísticas para probar la adecuación de la propuesta. Encuestas a los desarrolladores.
Fuente: Elaboración propia.		

2. Campo de Verificación

2.1 Ubicación Espacial

Los Desarrolladores que serán encuestados viven y laboran en la localidad del Departamento de Arequipa, provincia de Arequipa.

2.2 Ubicación Temporal

El desarrollo del presente trabajo se realiza en el presente entre los meses de noviembre de 2004 y junio de 2005.

2.3 Unidades de Estudio

Se ha decidido validar la adecuación de la propuesta de dos maneras teóricas:

- Basada en una investigación, comparación y discernimiento entre las características, procesos y estándares que utilizan los componentes COM+ para proveer transacciones y las características, procesos y estándares desarrollados de esta propuesta.

- Basada en la experiencia de los desarrolladores locales en base a las características, utilización de la Tecnología COM+ (En especial en el tema relacionado a la adecuación de las características transaccionales del modelo COM+) y respecto a la propuesta desarrollada. Se tomará como universo a las personas certificadas por Microsoft en el uso de la tecnología COM+, dicho número en la

actualidad es de 11 profesionales, siendo un número relativamente bajo se ha decidido no extraer una muestra de este universo, sino por el contrario trabajar con el universo entero.

3. Estrategia de Recolección de Datos

Se hará uso de las estrategias de observación y encuesta. A continuación se presenta la encuesta a utilizar:

La encuesta presentada a los 11 certificados Oficiales fue la siguiente:

Exposición del Modelo Transaccional COM+ para su Utilización En Actividades Transaccionales Todo o Nada Basadas en Servicios Web.

Encuesta para de determinación del grado de adecuación de la exposición respecto al trabajo transaccional de los componentes COM+.

Nombre:

_____

1) ¿Cómo considera la determinación del tiempo que debe manejar el DTC o el componente COM+ expuesto?
    a) Buena
    b) Muy buena
    c) Mala
    d) Muy mala

2) ¿Cómo considera que afectará la desactivación automática del componente expuesto luego de la llamada al método transaccional, no importando el resultado del mismo?
    a) contribuirá a la escalabilidad del aplicativo
    b) No contribuirá en nada a la escalabilidad del aplicativo

c) No tendrá ningún efecto sobre la performance del aplicativo participante

3) ¿Qué tan adecuada considera la implementación de la clase de fachada y su respectivo almacenamiento en las variables de aplicación a nivel del aplicativo participante?

a) Cumple con la infraestructura de coordinación y el protocolo durable de dos fases , además de permitir una implementación altamente escalable del aplicativo participante .

b) No cumple con la infraestructura de coordinación y el protocolo transaccional de dos fases , no permite la escalabilidad del aplicativo participante .

4) ¿Cómo considera las extensiones realizadas al protocolo WS-COORDINATION?

a) Necesarias contribuyen con un manejo adecuado de las actividades transaccionales.

b) No necesarias , no contribuyen al objetivo de coordinación de una actividad transaccional multiplataforma

5) ¿Cómo considera en líneas generales la Exposición del Modelo Transaccional COM+ para su Utilización En Actividades Transaccionales Todo o Nada Basadas en Servicios Web ?.

   a) Buena

   b) Muy buena

   c) Mala

   d) Muy mala

6) Indique por favor sus acotaciones o comentarios respecto al presente trabajo de investigación :

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

CRONOGRAMA

INFRAESTRUCTURA DE COORDINACIÓN

1.1 Introducción

En el presente capítulo se realizará el análisis y diseño de la arquitectura de software sobre la cual se basa la comunicación transaccional de los elementos participantes en una actividad Transaccional[35].

1.2 El problema

Todos los Aplicativos participantes en una actividad transaccional necesitan tener conocimiento de la existencia de los demás elementos participantes, para ello la propuesta de marco de trabajo transaccional WS-COORDINATION[36] define de forma parcial, la manera en la cual los participantes en una actividad transaccional establecen los enlaces de comunicación entre los miembros.

1.3 Solución

El autor propone el siguiente diagrama para la exposición del modelo transaccional COM+ en actividades transaccionales todo nada basadas en servicios Web

1.3.1. Contexto de la solución

Un Aplicativo Iniciador (APP-INI) desea invocar los métodos de un Servicio Web o aplicativo participante (APP-PAR), debe elegir un coordinador principal (COORD-PRI) para la actividad transaccional, una

[35] Ver significado de "Actividad Transaccional" en el glosario.

vez elegido el aplicativo iniciador le pide al COORD-PRI la creación de un contexto transaccional, denominado contexto transaccional inicial, para su utilización en la invocación al Servicio Web.

APP-INI invoca al Aplicativo Participante (APP-PAR) pasándole como parámetro el contexto transaccional inicial. APP-PAR elige un coordinador transaccional participante (COORD-PAR) para participar en la actividad transaccional, y le pide a COORD-PAR crear un contexto transaccional participante, pasándole cómo parámetro el contexto transaccional inicial.

APP-PAR, teniendo el contexto transaccional participante le pide al coordinador elegido COORD-PAR el registro de su participación en el coordinador principal COORD-PRI, pasándole para ello el contexto participante.

El coordinador participante COORD-PAR, utilizando la información del contexto participante se comunica con el coordinador principal COORD-PRI para pedir su registro en la actividad transaccional, enviando para ello la información del Servicio de Protocolo[37].

El coordinador principal COORD-PRI, registra al coordinador participante COORD-PAR, retornándole la información de su Servicio de Protocolo,

[36] Para más información referente a este estándar visitar http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-Coordination.pdf

[37] Para mayor detalle respecto al Servicio de Protocolo remítase al Marco Teórico del presente trabajo de Investigación.

de esta manera se tiende la línea de comunicación entre los coordinadores que administrarán la actividad transaccional.

1.3.2. Complementos al estándar WS-COORDINATION

Como primer complemento tenemos : el contexto transaccional no define cuantos coordinadores participantes va a administrar el coordinador principal COORD-PRI, por lo que es necesario ampliar tanto la petición de creación de contexto del coordinador, como los datos del contexto Transaccional, para reflejar esta información:

ESTRUCTURA DE TIPO :

CONTEXTO TRANSACCIONAL

WS-COORDINATION	PROPUESTA	TIPO DE DATO
Identificador de la Transacción	Identificador de la Transacción	String - GUID
Tiempo de Expiración	Tiempo de Expiración	Time
URL del Coordinador que Crea el Contexto	URL del Coordinador que Crea el Contexto	String
URL del Servicio de Registro	URL del Servicio de Registro	String
-	Número de Coordinadores Participantes	Integer
-	Contexto Participante	Contexto Transaccional
-	Fecha de Creación	Date Time

Fuente: Elaboración propia.

ESTRUCTURA DE TIPO :

TIEMPO

NOMBRE DE CAMPO	TIPO DE DATO	VALOR POR DEFECTO
Horas	Integer	0
minutos	Integer	0
Segundos	Integer	0

Fuente: Elaboración propia.

1.3.2.1 Descripción de las propiedades del tipo CONTEXTO TRANSACCIONAL

a) Identificador de la Transacción

Para el Identificador de la transacción se propone utilizar de preferencia un GUID[38], de esta manera se garantiza que el identificador de la transacción sea irrepetible.

b) Tiempo de Expiración

Es una estructura mediante la cual se indica en horas, minutos y segundos; el tiempo de duración de una transacción, partiendo de la hora de creación de la misma.

[38] Ver definición de GUID en el Anexo A: Glosario de Términos.

c) URL del Coordinador que Crea el Contexto

Contiene la URL del coordinador que creo el contexto transaccional.

d) URL del Servicio de Registro

Contiene la URL del Servicio de Registro perteneciente al Coordinador.

e) Número de Coordinadores Participantes

Contiene el número de Coordinadores Participantes que deberá considerar el Coordinador Principal, para que de esta manera él pueda aplicar los algoritmos necesarios para manejar el protocolo transaccional durable de dos fases.

f) Contexto Participante

Contiene una instancia del contexto participante, para el caso del Contexto transaccional inicial esa instancia contiene el valor nulo.

g) Fecha de Creación

Contiene la fecha de creación del contexto transaccional, expresado además en horas, minutos y segundos.

Como segundo complemento tenemos: Cuando el aplicativo Participante le pide al Coordinador Participante que registre su participación en la actividad transaccional adjunta el contexto transaccional, pero a la par adjunta información de su Interfaz Transaccional que contiene el URL que representa a la interfaz transaccional expuesta por APP-PAR para participar en el Protocolo Durable de Dos Fases (Complemento al protocolo WS-COORDIANTION). Esta complementación del estándar, permite enviar información de los aplicativos que llaman a COORD-PAR, y de esta manera, el Coordinador Participante tiene el punto de llamada para comunicar al Aplicativo Participante los mensajes del protocolo transaccional. La información que provee el Aplicativo Participante al Coordinador Participante se encuentra redefinida a continuación:

ESTRUCTURA DEL TIPO:

INFORMACION DE INTERFAZ TRANSACCIONAL

NOMBRE DE CAMPO	TIPO DE DATO	COMENTARIO
URL	String	Contiene la URL de la dirección de la interfaz transaccional expuesta por el Aplicativo Participante.

Fuente: Elaboración propia.

h) Información de Registro

ESTRUCTURA DEL TIPO : INFORMACIÓN DE REGISTRO

WS-COORDINATION	PROPUESTA	TIPO DE DATO
Identificador del protocolo	URL que define el tipo de protocolo transaccional soportado	String
URL del Servicio del Protocolo	URL a la cual fluirán los mensajes dependientes del protocolo	String

Fuente: Elaboración propia.

1.3.3 Casos de uso

Se consideran los siguientes casos de uso:

- Elegir el coordinador transaccional
- Crear la actividad Transaccional
- Participar en una actividad transaccional
- Registrar en una Actividad Transaccional existente

1.3.3.1. Diagrama de Casos de uso

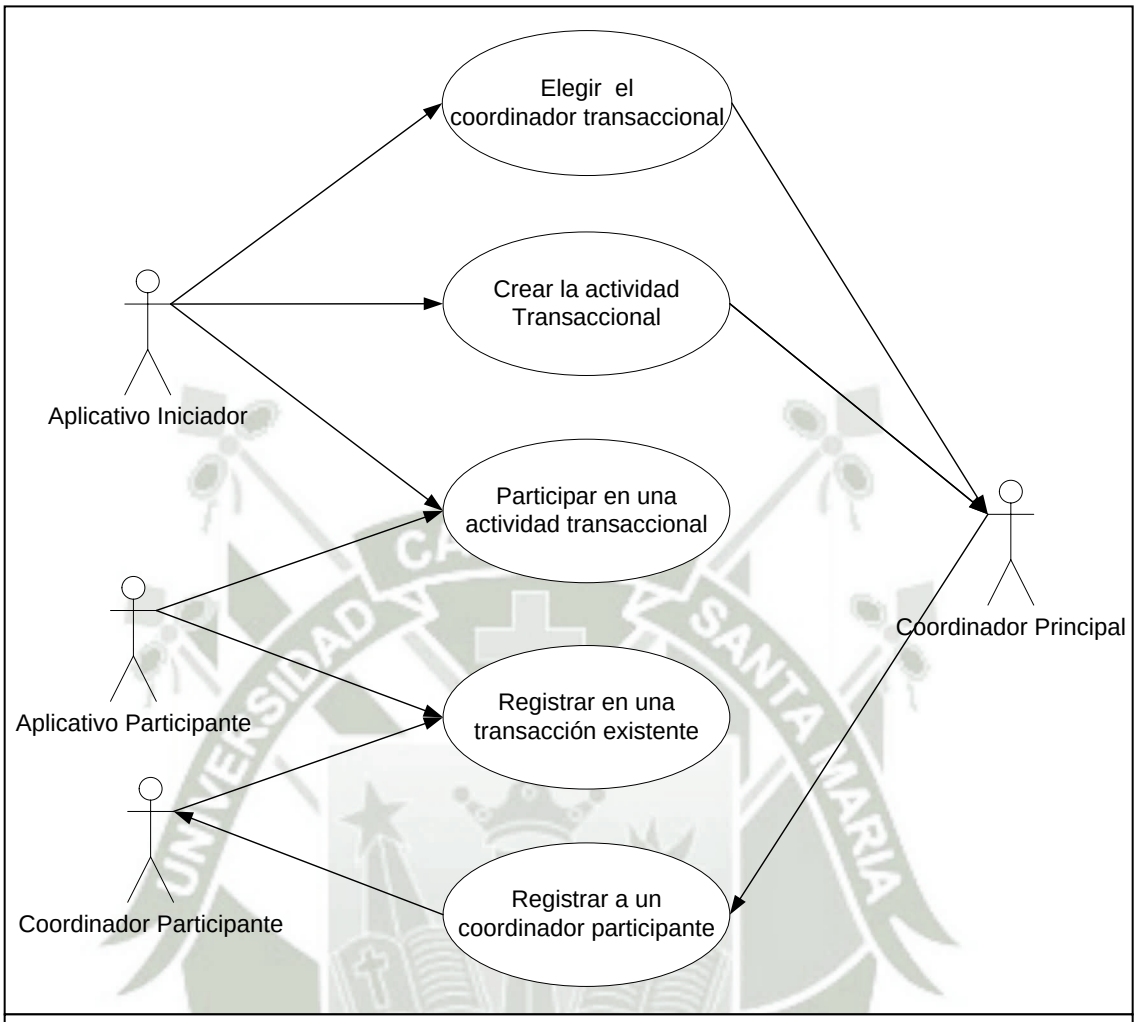


Diagrama 1 : Casos de Uso para la Infraestructura de Coordinación – Fuente : Elaboración propia.

1.3.3.2. Descripción de los casos de uso

Caso de Uso	Elegir el coordinador transaccional	CDU 01
Actores	Aplicativo Iniciador (APP-INI)	
Resumen	Este proceso permite descubrir y elegir un coordinador transaccional , para ser utilizado como coordinador principal .	
Tipo Caso de Uso	Primario	

Pre–Condiciones	El Aplicativo Iniciador tiene elementos de configuración dinámica ó estática para encontrar a un coordinador transaccional.

Flujo normal de eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario del Aplicativo Iniciador desencadena funcionalidad que para ser completada necesita el uso de un Aplicativo Participante.	
	2. El Aplicativo Iniciador elige el método de descubrimiento del coordinador.
	3 [si el método elegido en el punto 2 fue estático], el aplicativo iniciador rescata los datos necesarios para lograr la comunicación con el coordinador de la meta data estática.
	4. [si el método elegido en el punto 2 fue dinámico] el aplicativo iniciador utiliza el estándar UDII para encontrar al coordinador a ser utilizado en la operación, y extrae la información

	necesaria para realizar las comunicaciones entre al aplicativo iniciador y el coordinador principal. 5. El Aplicativo Iniciador inicia la Actividad Transaccional y cuando termina de forma exitosa utiliza los datos retornados por el Aplicativo Participante al que se llama para continuar con su proceso. 6. El aplicativo iniciador retorna el control al usuario
7. El usuario continua con la utilización del aplicativo iniciador.	

Caso de Uso	Crear la actividad Transaccional	CDU 02
Actores	Aplicativo Iniciador (APP-INI) Coordinador Transaccional Primario (COORD-PRI)	
Resumen	Este proceso permite crear un contexto transaccional principal por parte de un Aplicativo Iniciador de una Actividad Transaccional	
Tipo Caso de Uso	Primario	

Refer. Cruzada	CDU 01
Pre–Condiciones	El Aplicativo Iniciador conoce, tiene una referencia del Servicio Web que implementa el Coordinador que utilizará en el proceso transaccional, para ello el aplicativo iniciador debió realizar con éxito el caso de uso CDU 01.

Flujo normal de eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Aplicativo iniciador desea realizar diferentes llamadas a varios Aplicativos Participantes , por lo menos uno , de forma transaccional , para trabajar con un Coordinador Principal COORD-PRI , hace una llamada al coordinador de su elección. Pidiéndole la creación de un Contexto Transaccional Inicial, pasándole como parámetro el número de coordinadores participantes a utilizar .	2. El Coordinador COORD-PRI Crea un Contexto Transaccional, ver estructura de “Contexto Transaccional”. 3. El Coordinador almacena en su Lista de Transacciones activas la

	información del contexto que creo en el paso 2. 4. El Coordinador retorna el contexto transaccional creado al Aplicativo que invoca a la operación.
5. El Aplicativo iniciador realiza Varias llamadas a los Aplicativos Participantes, por lo menos una, adjuntando en cada llamada el contexto transaccional que se le devolvió en el paso 4.	

Caso de Uso	Participar en una actividad transaccional	CDU 03
Actores	Aplicativo Iniciador (APP-INI) Coordinador Transaccional Primario COORD-PRI Coordinador Transaccional Participante COORD-PAR Aplicativo Participante (APP-PAR)	
Resumen	Este proceso describe la forma en que un Aplicativo Participante puede participar en una actividad transaccional	
Tipo Caso de Uso	Primario	
Refer. Cruzada	CDU 01	

Flujo normal de eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Aplicativo participante (APP-PAR) recibe una petición para desencadenar funcionalidad expuesta, dentro de los parámetros que se le proveen se proporciona un Contexto Transaccional Inicial. Si este contexto transaccional existe, el aplicativo participante selecciona un coordinador Transaccional Participante COORD-PAR, para participar en la operación transaccional, una vez seleccionado el coordinador a utilizar, APP-PAR le pide a COORD-PAR, que	

cree un Contexto Transaccional Participante en la Actividad transaccional ya iniciada, pasando como parámetro el Contexto Transaccional Inicial.	2. El coordinador Participante COORD-PAR crea un contexto transaccional teniendo internamente una instancia del contexto transaccional principal. 3. El coordinador participante almacena en su lista de transacciones participantes tanto la información del Contexto Transaccional Principal como la información del Contexto Transaccional Participante y retorna el contexto participante al APP-PAR .

Caso de Uso	Registrarse en una transacción existente	CDU 04
Actores	Aplicativo Iniciador (APP-INI) Coordinador Transaccional Primario COORD-PRI Coordinador Transaccional Participante COORD-PAR Aplicativo Participante (APP-PAR)	
Resumen	Este proceso describe la forma en que un APP-PAR se registra como participante en una Actividad Transaccional existente por medio de su coordinador asociado COORD-PAR.	
Tipo Caso de Uso	Primario	
Refer. Cruzada	CDU 03	
Pre–condiciones	APP-PAR tiene una referencia del Coordinador COORD-PAR , que utilizará en el proceso transaccional. APP-PAR a Realizado el caso de uso CDU 03	

Flujo normal de eventos	
Acción del Actor	**Respuesta del Sistema**
1. APP-PAR luego de haber recibido el contexto Transaccional Participante, le pide a su coordinador asociado COORD-PAR que registre su participación en la actividad transaccional, dando parte al Coordinador Transaccional Principal, APP-PAR pasa en la petición de registro, el contexto	

participante generado en el CDU 03, además de ello le provee los datos de sus interfaz transaccional que participará en el protocolo transaccional de dos fases.	2. COORD-PAR verifica la existencia del contexto participante en su lista de transacciones participantes, si puede verificar la existencia utiliza los datos del servicio de registro del coordinador principal COORD-PRI para llamar a la operación de registro y guarda en los datos de la transacción participante , la instancia de la interfaz transaccional perteneciente al Aplicativo Participante. 3. El Coordinador principal realiza el Registro de un coordinador participante. 4. Se retorna al Coordinador Participante COORD-PAR información de registro[39], este la almacena en los datos de la transacción participante.

Para más información remitirse al documento ws-atomic transaction en el apendice.

[39] Ver el punto h) Información de registro, de la sección 3.3.2.1.Descripción de las propiedades del tipo CONTEXTO TRANSACCIONAL.

	De esta manera los coordinadores inmersos en una actividad transaccional se encuentran comunicados y preparados para el intercambio de mensajes dependientes de los protocolos transaccionales, como el durable de dos fases.

Caso de Uso	Registrar a un coordinador participante	CDU 05
Actores	Aplicativo Iniciador (APP-INI) Coordinador Transaccional Primario COORD-PRI Coordinador Transaccional Participante COORD-PAR Aplicativo Participante (APP-PAR)	
Resumen	Este proceso describe la forma en que un Coordinador Transaccional Principal registra en una actividad transaccional a un Coordinador Transaccional Participante.	
Tipo Caso de Uso	Primario	
Refer. Cruzada	CDU 04	
Pre–condiciones	Se realizó con éxito el caso de uso CDU 04	

Flujo normal de eventos	
Acción del Actor	**Respuesta del Sistema**
1. COORD-PAR le pide a COORD-PRI que lo registre como participante en una actividad transaccional, para tal efecto COORD-PAR proporciona una instancia de la información de su servicio de protocolo y una instancia del contexto participante.	2. COORD-PRI verifica la existencia del Contexto Transaccional Inicial, si este existe, verifica el cumplimiento de las condiciones de la actividad transaccional como la Propiedad Expires, si estas condiciones no son infringidas , COORD-PRI registra como participante a COORD-PAR en base a su información de Registro . 3. COORD-PRI retorna a COORD-PAR una instancia de la información de su Servicio de Protocolo.

1.3.4. Elección del medio de almacenamiento de la información Transaccional

Para almacenar la información que administra cada coordinador transaccional se proponen las siguientes Arquitecturas:

a) Manejo de información Transaccional por medio de Base de Datos

Con esta arquitectura los beneficios son inherentes a la administración y uso de base de datos, como es la seguridad, rapidez, disposición de diferentes modelos de explotación de la información transaccional, como ADO.NET, OLEDB, ODBC ADO, etc.

El uso de esta arquitectura plantea inconvenientes como la sobrecarga que conlleva la transformación de los mensajes SOAP a una estructura relacional u orientada a objetos según la elección de la base de datos, pero el inconveniente principal es la infracción a uno de los usos y fines de XML. La información se almacena en forma binaria y no puede ser comprendida por las personas sin una manipulación previa.

A continuación se presenta el diagrama de la arquitectura de esta propuesta:

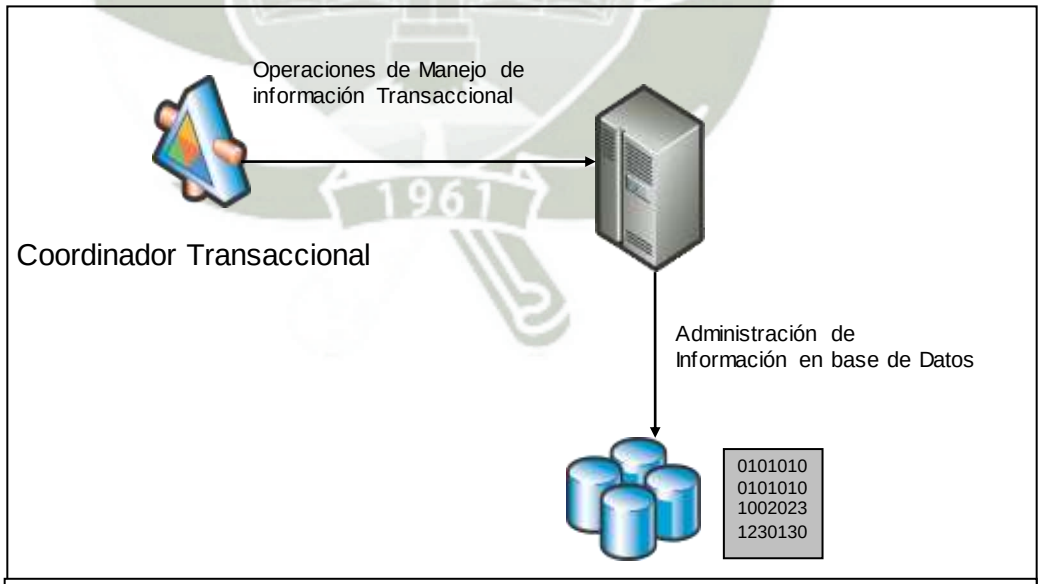


Figura 1: Manejo de información Transaccional por medio de Base de Datos.
Fuente: Elaboración propia.

b) Manejo de información Transaccional por medio archivos en disco y Windows Clustering

El manejo de la información transaccional por medio de archivos XML soportados en disco duro no infringe el uso del estándar XML, no trae sobrecarga por conversión a esquemas relacionales u orientados a objetos, y se puede beneficiar de tecnologías como Windows Clustering que permite tener arreglos de discos duros independientes, para diferentes servidores en el mismo cluster, de tal manera que los fallos se ven minimizados por la aplicación de redundancia de servidores, y de este modo provee una alta disponibilidad a los Coordinadores transaccionales que utilicen esta arquitectura.

Uno de los posibles inconvenientes para la aplicación de esta arquitectura es la sobrecarga causada al coordinador por la ejecución de un interpretador necesario para tener acceso a la información almacenada en los formatos SOAP soportados en documentos XML, este inconveniente se ve minimizado por la utilización de tecnologías como DOM que accede a datos XML en memoria y la salida al mercado de interpretadores (Parsers) cada vez más potentes.

A continuación se presenta el diagrama de esta arquitectura:

Manejo de información Transaccional por medio archivos en disco Y Windows Clustering

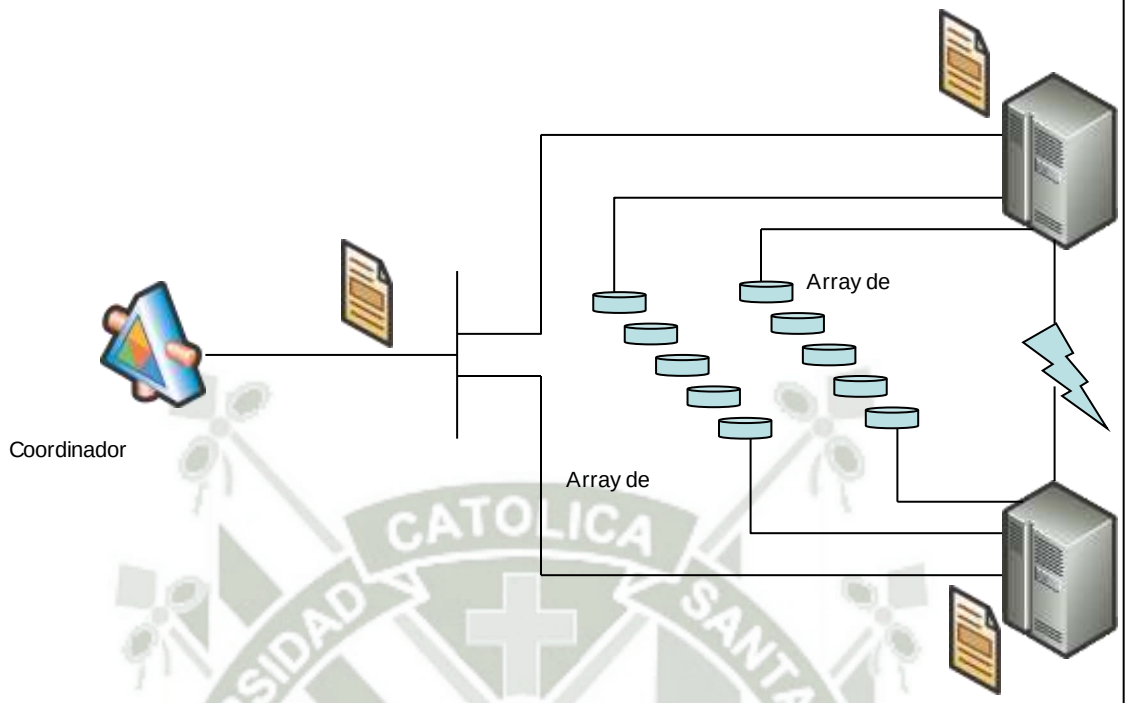


Figura 2: Manejo de información Transaccional por medio archivos en disco y Windows Clustering.
Fuente: Elaboración propia.

c) Estructura de directorios utilizada para el manejo de información transaccional

Para el manejo de información mediante archivos XML se propone la siguiente estructura de directorios:

- El nombre del directorio raíz que administra un coordinador se denominará “Coordinación”, al interior de esta carpeta se almacenará la información de todas las actividades transaccionales que el coordinador administre y todas las actividades transaccionales en las que el coordinador participe.

- Para manejar las actividades transaccionales administradas se creará una carpeta para cada una, con el identificador de dicha actividad transaccional inicial, al interior de esta carpeta se almacenará en un archivo XML el contexto transaccional inicial.

  Al interior de esta carpeta se encontrarán carpetas que representan a los coordinadores transaccionales participantes, los nombres de estas carpetas se corresponden con los identificadores de los contextos transaccionales participantes, aquí se almacena en un archivo XML los datos de contexto participante. Cada carpeta que representa a los contextos participantes tendrá interiormente tantas carpetas como protocolos transaccionales utilice el coordinador participante, al interior de esta carpeta se almacenarán en documentos XML los mensajes SOAP utilizados por ese protocolo.

- Para administrar la información sobre las actividades transaccionales en las que el coordinador participa se tiene como una directorio denominado “Participación“, este directorio es directorio hijo al directorio “Coordinación”, dentro del mismo para cada actividad transaccional en la cual participe el coordinador habrá una carpeta cuyo nombre será el identificador de la actividad transaccional inicial, e internamente a esta carpeta se encontrará tantas carpetas como protocolos participantes utilice el coordinador, al interior de cada uno de estos directorios se almacenan los mensajes que fluyen en base

al protocolo transaccional en formato XML, además de un documento XML que representa la información de la Interfaz Transaccional que provee el Aplicativo Participante para que se le notifique los mensajes que fluyen a través del protocolo transaccional .

A continuación se muestra un gráfico e la estructura de directorios que se administrará:

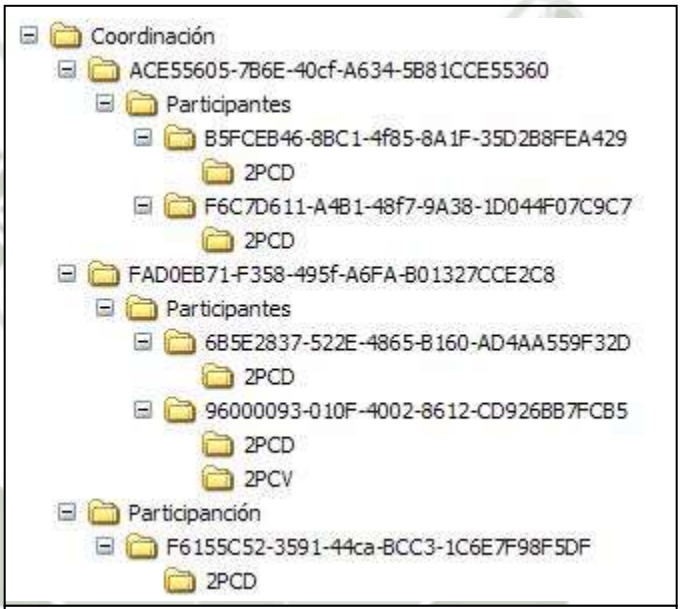


Figura 3: Estructura de Directorios.
Fuente: Elaboración propia.

1.3.5. Diagrama de Secuencia

En esta sección se muestran dos diagramas de secuencia, el primero de los casos de uso: CDU 01, CDU 02 y CDU 03. El segundo de los casos de uso CDU 04 y CDU 05.

Diagrama de Secuencia Casos de uso CDU 01, CDU 02 y CDU 03

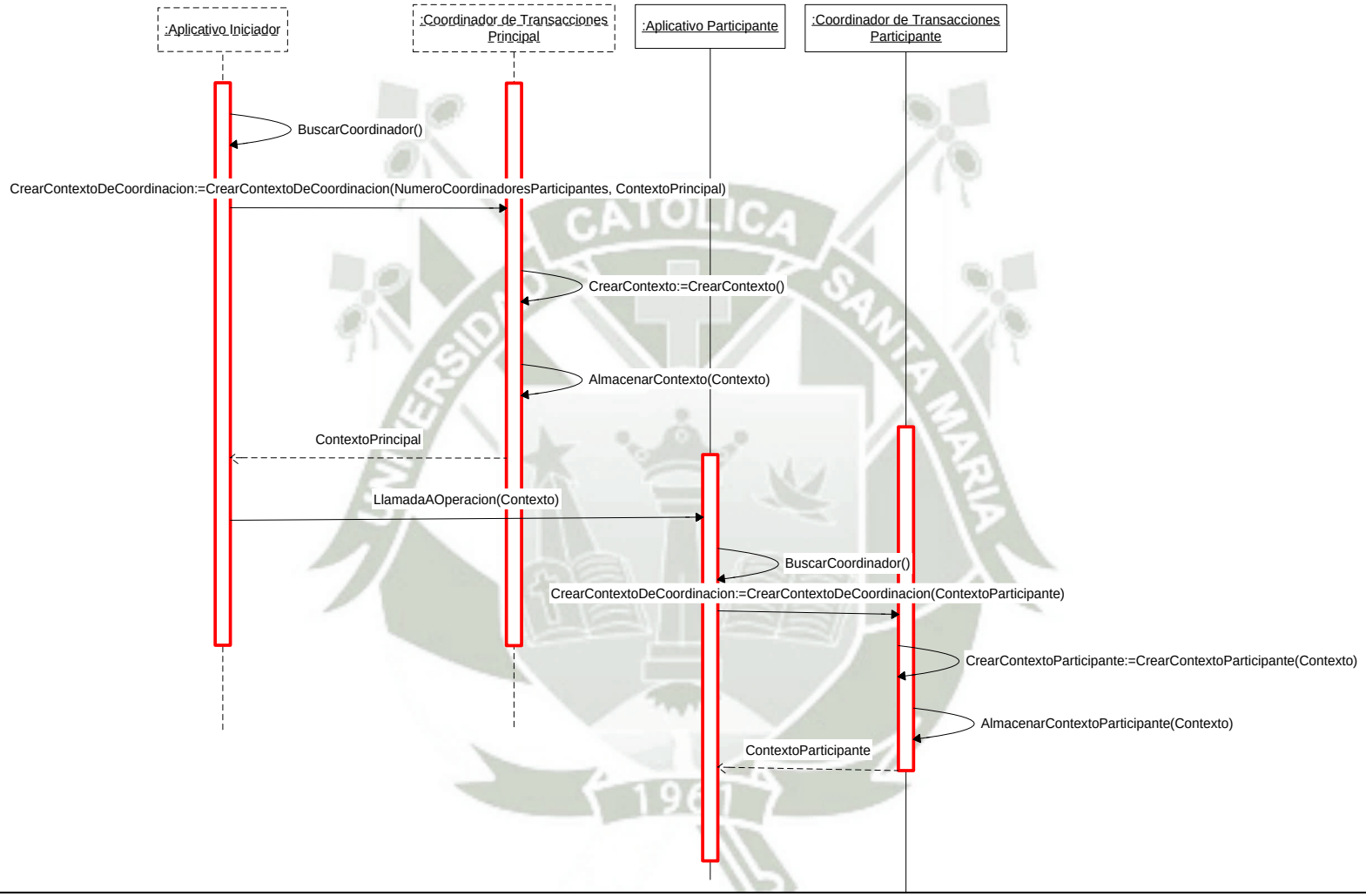


Diagrama 2: Diagrama de secuencia de Casos de uso CDU01, CDU02, y CDU03. Fuente: Elaboración propia.

Diagrama de Secuencia Casos de uso CDU 04, CDU 05

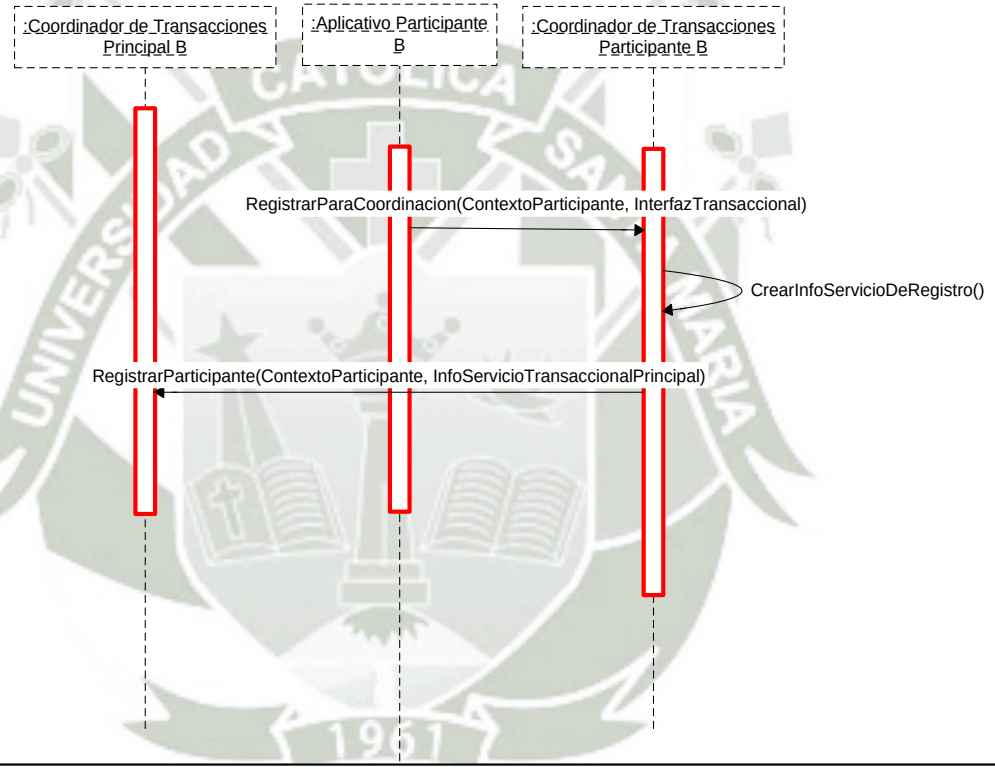


Diagrama 3: Diagrama de secuencia de Casos de uso CDU04, y CDU05. Fuente: Elaboración propia.

1.3.6. Diagrama de Clases

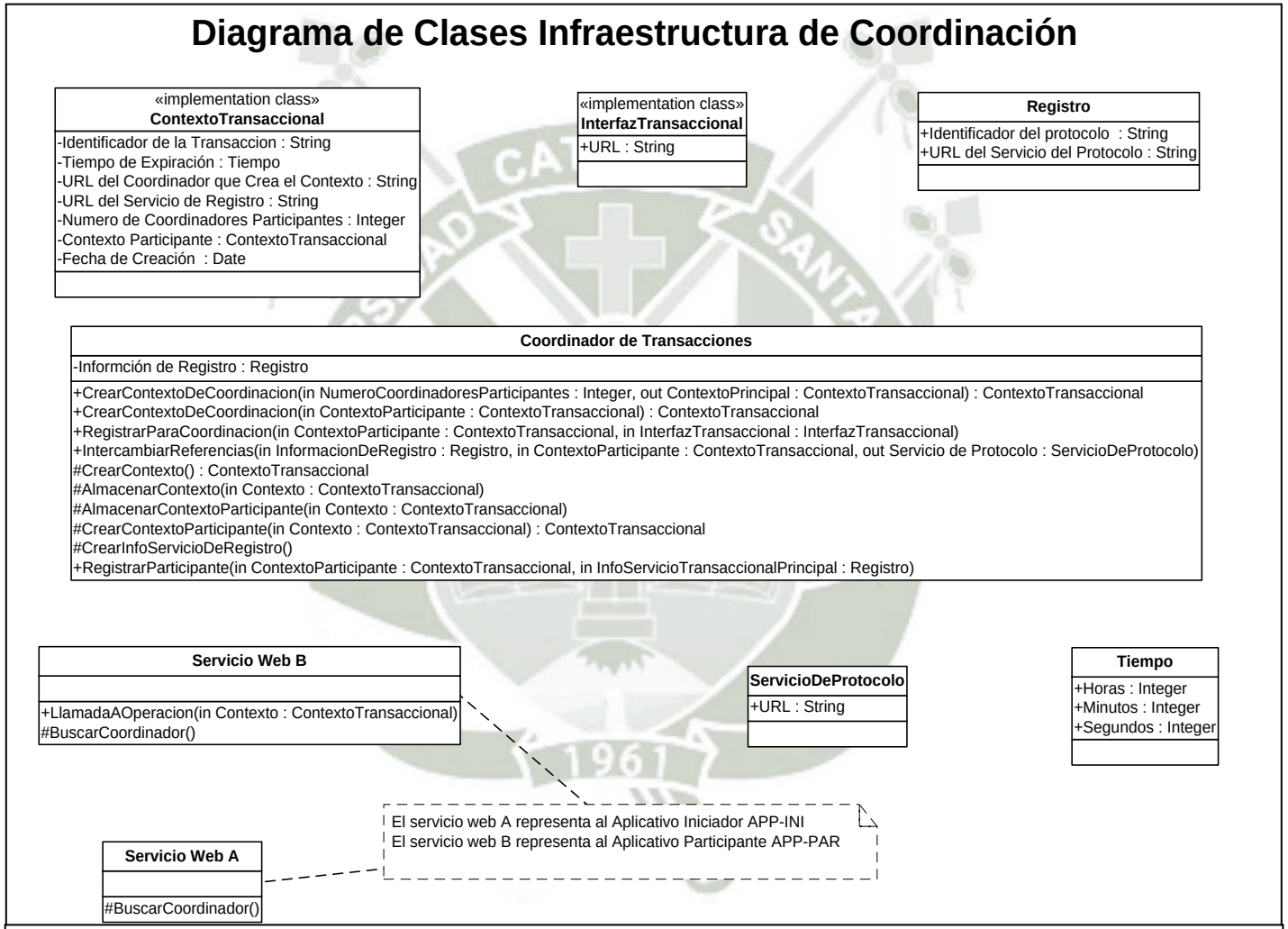


Diagrama 4: Diagrama de clases de Infraestructura de coordinación. Fuente: Elaboración

1.3.7. Pseudocódigo

A continuación se muestra el Pseudocódigo de la implementación de la Infraestructura de Coordinación.

a) Contexto transaccional

```
' Static Model
Public Class ContextoTransaccional
   Private IdentificadordelaTransaccion As String
   Private TiempodeExpiración As Tiempo
   Private URLdelCoordinadorqueCreaelContexto As String
   Private URLdelServiciodeRegistro As String
   Private NumerodeCoordinadoresParticipantes As Integer
   Private ContextoParticipante As ContextoTransaccional
   Private FechadeCreación As Date
End Class ' END CLASS DEFINITION ContextoTransaccional
```

b) Coordinador de Transacciones

```
' Static Model
Private Class CoordinadordeTransacciones
   Private InformcióndeRegistro As Registro
   Public Overridable Function CrearContextoDeCoordinacion
   (ByVal NumeroCoordinadoresParticipantes As Integer, _
   ByRef ContextoPrincipal As ContextoTransaccional) As
   ContextoTransaccional
         Instanciar un objeto de la clase contexto
         transaccional
         llenar las propiedades adecuadas como :
         - Tiempo de expiración
         - Dirección de El coordinador
```

```
End Function

Public Overridable Function CrearContextoDeCoordinacion
(ByVal ContextoParticipante As ContextoTransaccional) As
ContextoTransaccional
      Crear el contexto transaccional
      setear internamente la referencia del contexto
      participante a  la instancia que se pasa
      de ene l parámetro "ContextoParticipante"
End Function

Public Sub RegistrarParaCoordinacion (ByVal
ContextoParticipante As ContextoTransaccional, _
ByVal InterfazTransaccional As InterfazTransaccional)
      Verificar la participación en la actividad
      transaccional
      - verificar la existencia del contexto participante
      - almacenar la interfaz transaccional en la
        información
        de las actividades participantes.
End Sub

Public Sub IntercambiarReferencias (ByVal
InformacionDeRegistro As Registro, ByVal
ContextoParticipante As ContextoTransaccional, ByRef
ServiciodeProtocolo As ServicioDeProtocolo)
      El coordinador principal deberá abrir
      la instancia del contexto participante y extraer
      la instancia de el contexto transaccional inicial
      para verificar en base a su identificador la
```

```
        información de actividad transaccional.
        almacenar la información del servicio de protocolo
End Sub

Public Sub RegistrarParticipante (ByVal
ContextoParticipante As ContextoTransaccional, _
ByVal InfoServicioTransaccionalPrincipal As Registro)
        Verificar la validez de la transacción
        participante para la cual se pide el registro
        en base a la información de el contexto participante
        Utilizar  la información de el contexto transaccional
        principal para obtener la información del servicio de
        registro y preparar la petición de registro
        al coordinador principal.
        Almacenar la información de la interfaz transaccional
        expuesta por el aplicativo participante , en los
        datos de la transacción participante
End Sub

Protected Function CrearContexto () As
ContextoTransaccional
        Instanciar un obejto de la clase contexto
        transaccional
        Llenar las propiedades adecuadas como :
        Identificador de la Transacción
        Tiempo de Expiración
        URL del Coordinador que Crea el Contexto
        URL del Servicio de Registro
        Número de Coordinadores Participantes
        Contexto Participante
```

```
      Fecha de Creación
End Function

Protected Sub AlmacenarContexto (ByVal Contexto As
ContextoTransaccional)
      Tomando información de metadata
      acerca del recurso que se utilizapara
      almacenar la información transaccional
      instanciar los objetos adecuados como
      strems soap y grabar el mensaje soap
End Sub

Protected Sub AlmacenarContextoParticipante (ByVal
Contexto As ContextoTransaccional)
      Realizar los mismos pasos que en el
      método aolmacenar contexto , pero
      en este caso la metadata debe ser adecuada
      a la grabación de un contexto transaccional
      incial
End Sub

Protected Function CrearContextoParticipante (ByVal
Contexto As ContextoTransaccional) As
ContextoTransaccional
      Realizar los mismos pasos que el método crear
      contexto , en este caso setear la propiedad interna
      que representa al contexto participante con
      el parámetro "contextoParticipante"
End Function
```

```
   Protected Sub CrearInfoServicioDeRegistro ()
         Instanciar un objeto de la clase Registro
         y llenar sus parámetros :
         URL que define el tipo de protocolo transaccional
         soportado
         URL a la cual fluirán los mensajes dependientes del
         protocolo
         Utilizando la metadata  de el coordinador
   End Sub

End Class ' END CLASS DEFINITION CoordinadordeTransacciones
```

c) Interfaz Transaccional

```
' Static Model
Public Class InterfazTransaccional
   Public URL As String
End Class ' END CLASS DEFINITION InterfazTransaccional
```

d) Registro

```
' Static Model
Public Class Registro
   Public Identificadordelprotocolo As String
   Public URLdelServiciodelProtocolo As String

End Class ' END CLASS DEFINITION Registro
```

e) Servicio de Protocolo

```
' Static Model
Public Class ServicioDeProtocolo
```

```
   Public URL As String
End Class ' END CLASS DEFINITION ServicioDeProtocolo
```

f) Servicio Web A

```
' Static Model
Private Class ServicioWebA
   Protected Sub BuscarCoordinador ()
         Definir el método configurado en este
         Aplicativo iniciador para  encontrar a su coordiandor
         principal , ejecutar este método para
         encontrar al coordinador participante .
   End Sub
End Class ' END CLASS DEFINITION ServicioWebA
```

g) Servicio Web B

```
'Static Model
Private Class ServicioWebB
   Public Sub LlamadaAOperacion (ByVal Contexto As
   ContextoTransaccional)
         En esta llamada llamar al coordinador transaccional
         principal para que cree el contexto transaccional
         inicial , realizar la llamada al aplicativo
         participante , llamar al coordinador principal
         para preguntar por el estado de la actividad
         transaccional .
   End Sub
   Protected Sub BuscarCoordinador ()
         realizar los mismos  pasos que toman lugar en la
         clase Servicio Web A
```

```
    End Sub
End Class ' END CLASS DEFINITION ServicioWebB
```

h) Tiempo

```
' Static Model
Public Class Tiempo
    Public Horas As Integer
    Public Minutos As Integer
    Public Segundos As Integer
End Class ' END CLASS DEFINITION Tiempo
```

1.4. Resumen

En este capítulo se pudo ver los complementos al estándar WS-COORDINATION, necesarios para poder implementar los elementos participantes en una actividad transaccional.

Vimos además los casos de uso de sistema que delinean el comportamiento de la infraestructura transaccional.

La discusión de los métodos de almacenamiento de la información transaccional por parte de los coordinadores transaccionales brinda una discusión ilustrativa sobre los beneficios de utilizar archivos serializados a disco o bases de datos transaccionales.

Finalmente los diagramas de secuencia, de clases y el pseudocódigo de los elementos participantes nos brindan una idea amplia de la

interacción entre los elementos participantes en una actividad transaccional, además de un panorama amplio para la implementación de estos elementos.

EL PROTOCOLO DURABLE DE DOS FASES

1.1 Introducción

El Marco de trabajo WS-TRANSACTION define el Protocolo Durable de dos Fases y los mensajes que fluyen en la interacción de los participantes en una actividad transaccional, además de los estados de los participantes que son resultado de la comunicación por medio de los mensajes que fluyen a través del protocolo.

En este capítulo complementaremos el diseño realizado en el capitulo INFRAESTRUCTURA DE COORDINACIÓN, para delinear el comportamiento de los elementos participantes, en especial el de los coordinadores transaccionales, tanto el Coordinador Transaccional Principal, como los Coordinadores Transaccionales Participantes.

1.2 El problema

Una vez logrado el enlace de comunicación entre el Coordinador Transaccional Principal COORD-PRI y el Coordinador Transaccional Participante COORD-PAR, la comunicación entre ambos puede proceder como parte del protocolo transaccional.

El coordinador Transaccional Principal empieza el protocolo de durable de dos fases para coordinar la actividad transaccional.

1.3 Solución

1.3.1 Contexto de la solución

El coordinador Transaccional Principal empieza la coordinación de la actividad transaccional cuando el número de Coordinadores transaccionales registrados en la actividad es igual al número de coordinadores que fue enviado por el Aplicativo Iniciador en su petición de creación del contexto transaccional inicial.

Para cada uno de los coordinadores Participantes el Coordinador principal , empieza el manejo del protocolo durable de dos fases, enviando el mensaje “Prepare” , los coordinadores obtienen los recursos necesarios para realizar sus operaciones transaccionales, una vez que todos ellos han podido realizar esta operación responden al coordinador principal con el mensaje “Prepared”, cuando el coordinador obtiene el mensaje “Prepared” de todos los coordinadores participantes, el Coordinador Transaccional Principal , puede enviar a cada uno de ellos el mensaje “Commit” para que los cambios realizados en los recursos que administra cada Coordinador Participante sean modificados de forma permanente, los coordinadores participantes pueden responder con el mensaje “Commited” cuando todos los coordinadores participantes han respondido con este mensaje la actividad transaccional se da por terminada de forma Adecuada.

Cuando un Coordinador Participante no puede Preparar o Comitir una transacción envía el mensaje “Aborted”, el manejo completo de los estados del protocolo durable de dos fases se encuentra detallado en el protocolo WS-ATOMIC TRANSACTION referencia al apéndice donde se encuentran las tablas.

1.3.2 Casos de Uso

Aquí se muestran los casos de uso más significados para llevar a cabo el protocolo durable de dos fases.

- Iniciar el protocolo transaccional durable de dos fases.
- Comunicar los mensajes en la actividad transaccional.
- Comitir La actividad transaccional (no presentaremos su diagrama de estados puesto a que posee la misma representación que el caso de uso “Comunicar los mensajes en la actividad transaccional”).
- Terminar la actividad transaccional.

1.3.2.1 Casos de Uso

A continuación se muestra en esta sección el diagrama de casos de uso seguido de la descripción de casos de uso.

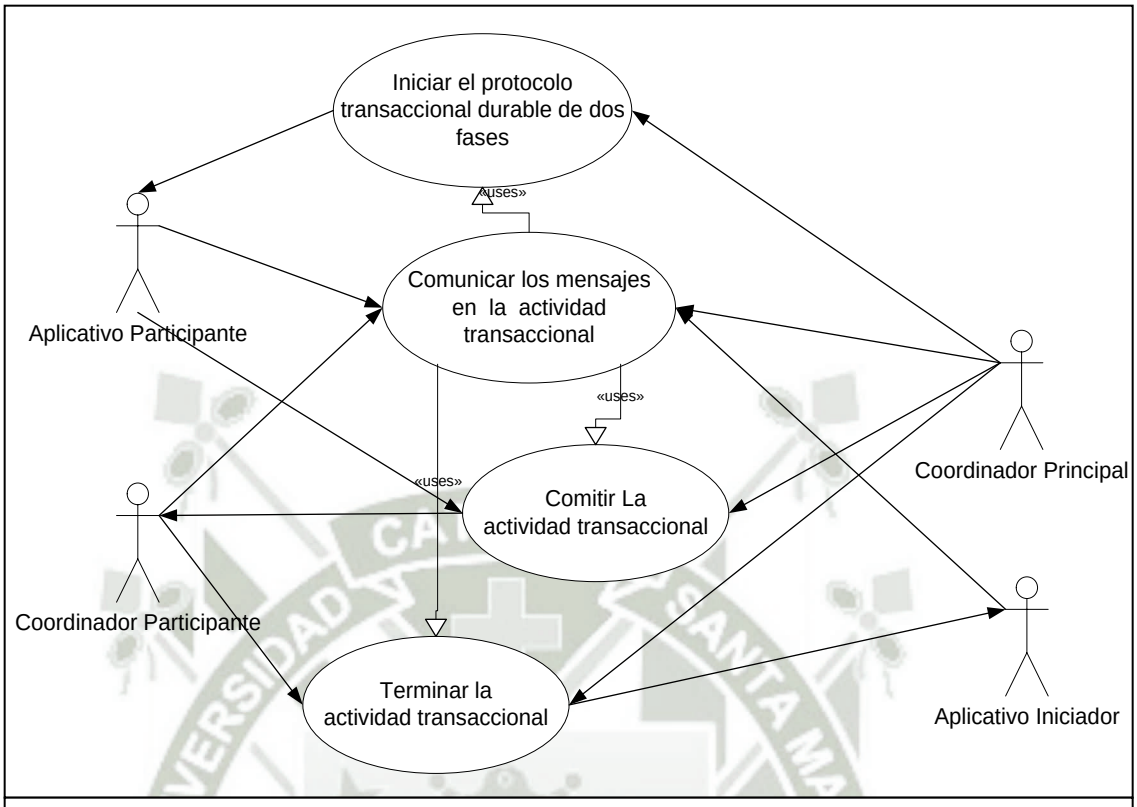


Diagrama 4: Diagrama de Casos de Uso para El protocolo durable de dos Fases. Fuente: Elaboración propia.

1.3.2.2 Descripción de los casos de uso

Caso de Uso	Iniciar el protocolo transaccional durable de dos fases	CDU 06
Actores	– Aplicativo Iniciador (APP-INI) – Coordinador Transaccional Primario COORD-PRI – Coordinador Transaccional Participante COORD-PAR - Aplicativo Participante (APP-PAR)	

Resumen	Este proceso describe la forma en que un Coordinador Transaccional Principal inicia el protocolo transaccional durable de dos fases, este proceso representa la primera fase del protocolo transaccional.
Tipo Caso de Uso	
Refer. Cruzada	
Pre–condiciones	

Flujo normal de eventos	
Acción: del Actor(es)	**Acción: del Sistema**
1. Los Coordinadores Transaccionales Participantes se registran con el coordinador transaccional Principal , como participantes en la Actividad Transaccional .	2. COORD-PRI que se encuentra monitoreando la información de las transacciones activas determina que el número de Coordinadores transaccionales participantes es igual al número de coordinadores que se paso en el evento de creación del contexto transaccional inicial .

	3. COORD-PRI Comienza a utilizar la información de cada coordinador participante registrado para comenzar a enviar el mensaje “PREPARE” , adjuntando al mensaje el contexto transaccional inicial . 4. Se almacena la información de los mensajes enviados a cada uno de los coordinadores participantes
5. El coordinador Participante comprueba la participación en la actividad transaccional por medio del contexto transaccional inicial . 6. El coordinador participante le envía al aplicativo participante el mensaje, si el Aplicativo participante puede preparar la actividad transaccional , responde al coordinador participante con un mensaje “Prepared”.	

7. El coordinador participante envía este mensaje al Coordinador transaccional Principal.	8. El coordinador Transaccional Principal almacena en la información perteneciente al Coordinador Transaccional Participante el mensaje enviado en el paso 7.

Caso de Uso	Comunicar los mensajes en la actividad transaccional	[CDU 07]
Actores	– Aplicativo Iniciador (APP-INI) – Coordinador Transaccional Primario COORD-PRI – Coordinador Transaccional Participante COORD-PAR Aplicativo Participante (APP-PAR)	
Referencia REF		
Resumen		
Tipo Caso de Uso	Este proceso describe la forma en que los coordinadores transaccionales participantes hacen participe a los Aplicativos Participantes acerca de los mensajes que Fluyen por el Protocolo Durable de Dos Fases y Viceversa.	

Refer. Cruzada	
Pre–condiciones	
Flujo normal de eventos	
Acción: del Actor(es)	**Acción: del Sistema**
1. El caso de uso se inicia cuando un Elemento participante (Remitente) en una actividad transaccional empieza el proceso de envío de un mensaje transaccional a otro elemento (Destinatario).	2. El Remitente debe buscar la dirección a la cual se envía el mensaje , para el caso de los mensajes transaccionales que fluyen a través del protocolo durable de dos fases esta información se encuentra en la instancia del servicio del protocolo que todos los coordinadores inmersos en la actividad transaccional poseen . 3. El Remitente instancia un objeto de la clase encargada de armar los mensajes del protocolo durable de dos fases.

	4. El remitente instancia un objeto de la clase encargada de la mensajería entre elementos participantes en la actividad transaccional. 5. El remitente envía el mensaje al Destinatario.

Caso de Uso	Comitir la Actividad transaccional	[CDU 08]
Actores	- Coordinador Transaccional (COORDA , COORDB) - Aplicativo Participante (AP)	
Referencia REF		
Resumen	Este proceso describe la finalización de las transacciones durables de dos fases ,se explica la segunda fase del protocolo durable de dos fases.	
Tipo Caso de Uso		
Refer. Cruzada		
Pre–condiciones		

Flujo normal de eventos	
Acción: del Actor(es)	**Acción: del Sistema**
1. El Coordinador Principal COORDA una vez que tiene el mensaje “PREPARED” de todos los coordinadores participantes, puede enviar a cada uno de ellos el mensaje “COMMIT”, el coordinador principal almacena el envío de este mensaje en los datos de la actividad transaccional.	2. El coordinador Participante recibe el mensaje “COMMIT” que viene acompañado con la información del contexto transaccional inicial , este revisa su lista de transacciones participantes , si existe una transacción participante en la actividad transaccional por la cual se recibe el mensaje “COMMIT”, se almacena el mensaje enviado al coordinador participante en los datos de la transacción participante .

	3. El coordinador participante hace una llamada al método commit de la aplicación participante, si este no retorna ningún error , se retorna al coordinador principal el mensaje "COMMITED" , el coordinador participante almacena la información de este mensaje en los datos de la actividad transaccional

Caso de Uso	Terminar la Actividad transaccional	[CDU 09]
Actores	- Coordinador Transaccional (COORDA , COORDB) - Aplicativo Participante (AP)	
Referencia REF		
Resumen		
Tipo Caso de Uso		
Refer. Cruzada		
Pre–condiciones		

Flujo normal de eventos	
Acción: del Actor(es)	**Acción: del Sistema**
1. El Coordinadores Participantes envían el mensaje “COMMITED” al Coordinador Transaccional Principal, cuando el Aplicativo Participante a confirmado la operación sobre el recurso transaccional.	
	2. El Coordinador principal Almacena esta información adjunta a la información del contexto transaccional inicial. Cuando el Aplicativo Iniciador Pregunta por el resultado de la Actividad Transaccional , la información se rescata y se envía al Aplicativo Iniciador

1.3.3 Diagrama de Secuencia

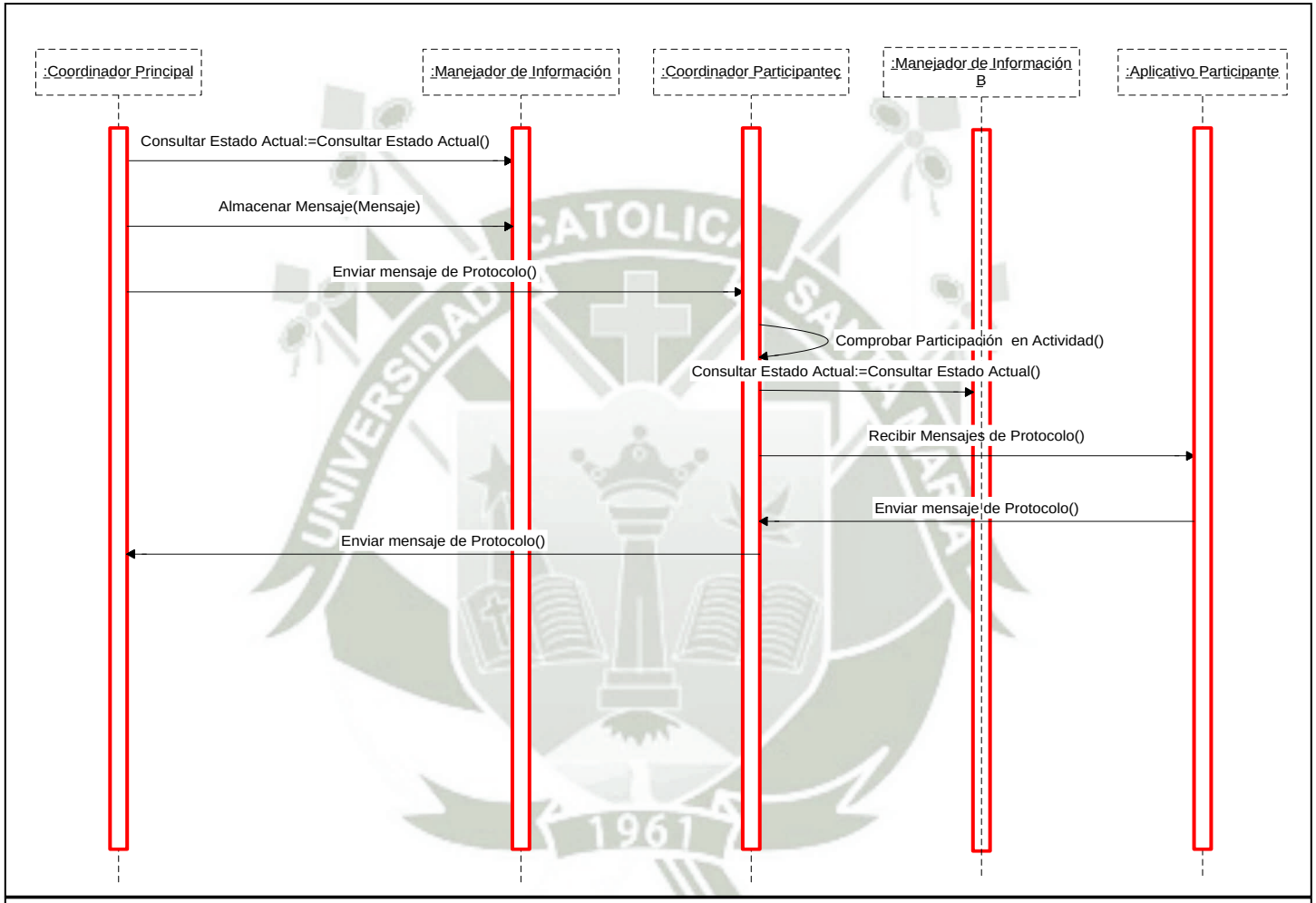


Diagrama 5: Diagrama de Secuencia Casos de uso CDU 06. Fuente: Elaboración propia.

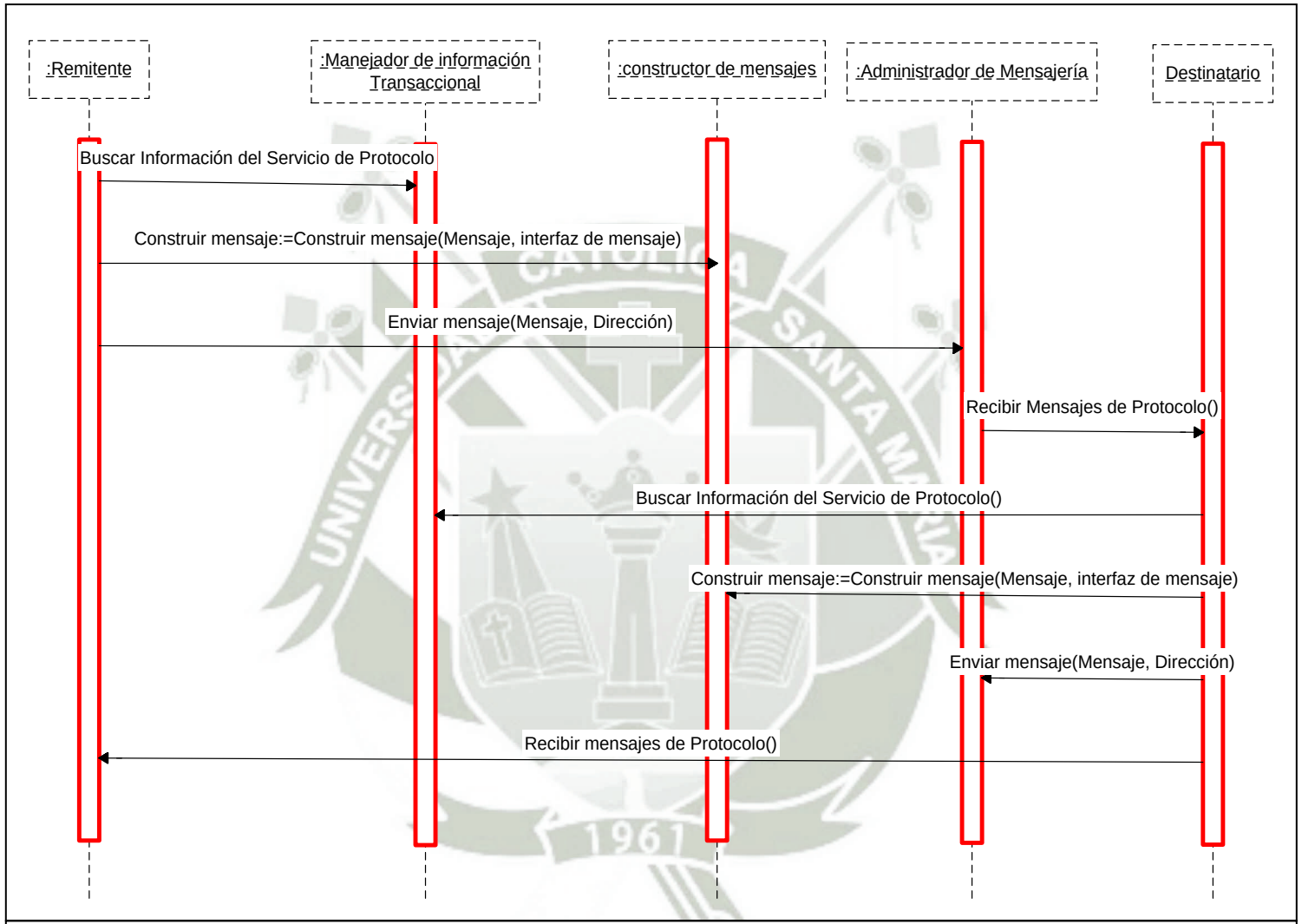


Diagrama 6: Diagrama de Secuencia Casos de uso CDU 07. Fuente: Elaboración propia.

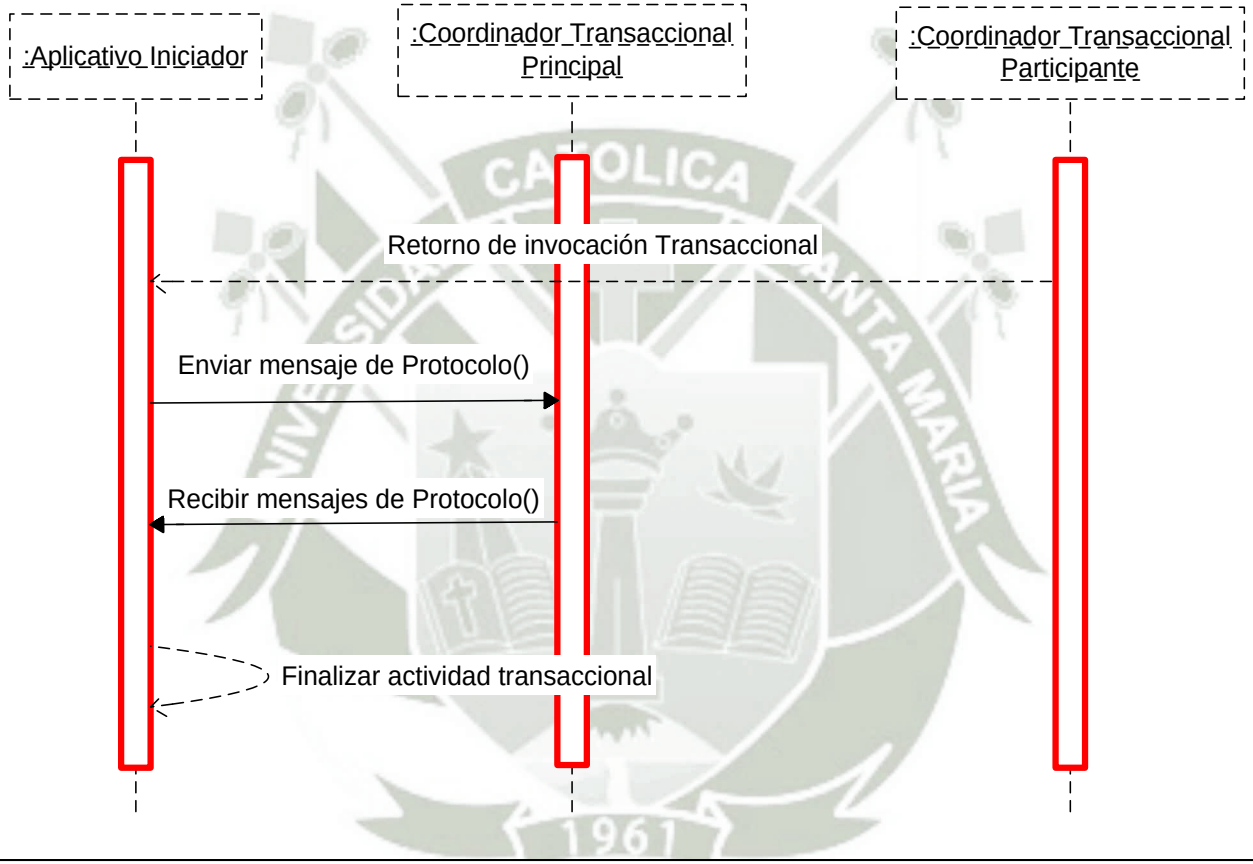


Diagrama 7: Diagrama de Secuencia Casos de uso CDU 09. Fuente: Elaboración

1.3.4 Diagrama de Clases

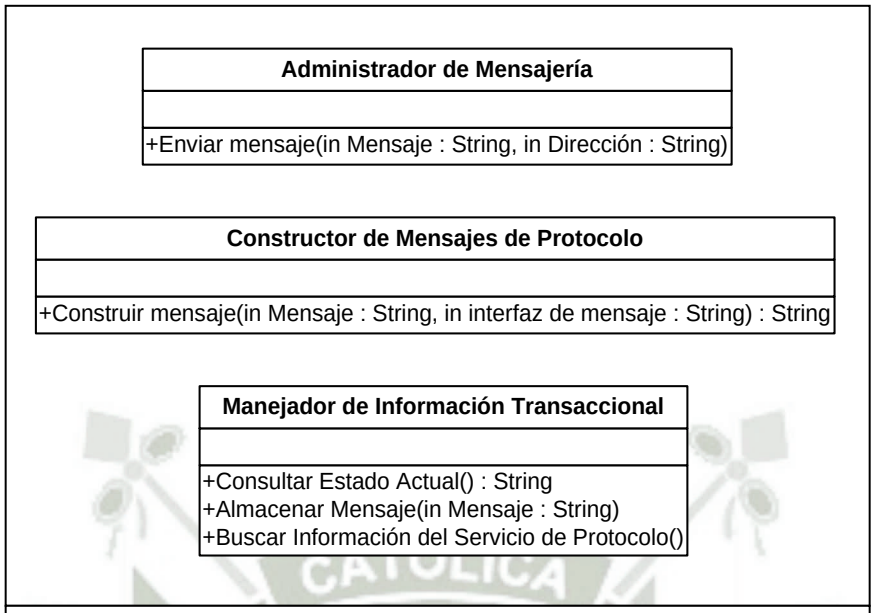


Diagrama 8: Diagrama de Clases complementarias a los elementos transaccionales Participantes en una Actividad Transaccional. Fuente: Elaboración propia.

1.3.5 Pseudocódigo

a) Administrador de Mensajería

```
' Static Model

Public Class AdministradordeMensajería

   Public Sub Enviarmensaje (ByVal Mensaje As String, _
                                 ByVal Dirección As String)
         Este procedimiento es encargado de enviar el
         mensaje al destinatario , para tal fin se provee
         el siguiente ejemplo de código en visual basic
         .net
         Dim s As Stream
         Dim req As HttpWebRequest =
         WebRequest.Create(direccion del servicio de
         protocolo)
         añadir el contenido a la petición
```

```
            s = req.GetRequestStream()
            examinar el resultado de la petición si es necesario
        End Sub
    End Class ' END CLASS DEFINITION AdministradordeMensajería
```

b) Constructor de Mensajes de Protocolo

```
' Static Model
Public Class ConstructordeMensajesdeProtocolo
    Public Function Construirmensaje (ByVal Mensaje As String,
    ByVal interfazdemensaje As String) As String
    comenzar a construir un flujo de caracteres
    -------------------------------
    Dim st As MemoryStream = New MemoryStream(1024)
    Dim result As String
    Dim buffer() As Byte

    construir el sobre SOAP
    -------------------------------
    Dim tr As New XmlTextWriter(st, Encoding.UTF8)
    tr.WriteStartDocument()
    tr.WriteStartElement("soap", "Envelope",
"http://schemas.xmlsoap.org/soap/envelope/")
    tr.WriteAttributeString("xmlns", "xsi", Nothing,
"http://www.w3.org/2001/XMLSchema-instance")
    tr.WriteAttributeString("xmlns", "xsd", Nothing,
"http://www.w3.org/2001/XMLSchema")
    tr.WriteAttributeString("xmlns", "soap", Nothing,
"http://schemas.xmlsoap.org/soap/envelope/")
```

```
   tr.WriteStartElement("Body",
"http://schemas.xmlsoap.org/soap/envelope/")
   empezar a construir el mensaje según su esquema
   ---------------------------------
   tr.WriteStartElement(Nothing, "GetAccount",
"http://tempuri.org/")
   tr.WriteElementString("acctID", "1")
   tr.WriteEndElement()
   tr.WriteEndElement()
   tr.WriteEndDocument()
   tr.Flush()
   buffer = st.GetBuffer()
   Dim d As Decoder = Encoding.UTF8.GetDecoder()
   Dim chars() As Char
   ReDim chars(buffer.Length)
   d.GetChars(buffer, 2, buffer.Length - 2, chars, 0)
   result = New String(chars)
   tr.Close()
   st.Close()
   en la varible resultado se encuentra el sobre SOAP
   Colocar a final las cabeceras al mensaje
   a continuación se da un ejemplo de un sobre SOAP
   completo :
   ------------------------------------------------------------
   POST /dummy/service1.asmx HTTP/1.1
         Host: 192.168.0.80
         Content-Type: text/xml; charset=utf-8
         Content-Length: 215
         SOAPAction: "http://woodgrovebank.com/GetAccount"
         <?xml version="1.0" encoding="utf-8"?>
```

```
        <soap:Envelope xmlns:xsi="http://www.w3.org/2001/?
                    XMLSchema-instance"?
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"?
      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
          <soap:Body>
            <GetAccount xmlns="http://woodgrovebank.com">
              <acctNumber>1234</acctNumber>
            </GetAccount>
          </soap:Body>
        </soap:Envelope>
    End Function
  End Class ' END CLASS DEFINITION
  ConstructordeMensajesdeProtocolo
```

c) Manejador de Información Transaccional

```
Public Class ManejadordeInformaciónTransaccional
    Public Function ConsultarEstadoActual () As String
          buscar la información del estado actual en base a
          a mensajes anteriores y mensaje actual de el
          coordinador , ver tabla de estados y mensajes
          de el protocolo transaccional durable de dos
          fases en los apendices
    End Function

    Public Sub AlmacenarMensaje (ByVal Mensaje As String)
          Almacenar mensaje ,
          si se trata de un almacenamiento en archivos
          se puede proceder de la siguiente forma
          Dim Dir As String = Directorio
```

```
            Dim stream As Stream = File.Open(Dir & _
                        NombreArchivo & ".xml", FileMode.Create)
            Dim formatter As New SoapFormatter
            formatter.Serialize(stream, Objeto)
            stream.Close()
            el ejemplo de código anterior guardará en un archivo
            XML  el mensaje completo .
        End Sub

        Public Sub BuscarInformacióndelServiciodeProtocolo ()
            Buscar en la información del coordinador principal
            o del coordinador participante , la instancia del
            servicio de protocolo que trae la información sobre
            el punto de destino de los mensajes transaccionales
        End Sub

End Class ' END CLASS DEFINITION
ManejadordeInformaciónTransaccional
```

1.4 Resumen del Capítulo

En este capítulo hemos visto la forma el la cual los elementos participantes en una actividad transaccional pueden llevar a cabo el protocolo transaccional durable de dos fases, no se amplió el estándar WS-ATOMIC TRANSACTION, se creó la interfaz transaccional de los aplicativos participantes, para que estos puedan recibir los mensajes transaccionales de los Coordinadores transaccionales Participantes.

Creamos la recomendación de comunicación desacoplada entre los participantes a partir de dos clases:

- Constructor de Mensajes de Protocolo encargada de construir los mensajes transaccionales

- Administrador de Mensajería, encargado del envío de los mensajes transaccionales a un aplicativo participante

Manejador de Información Transaccional: encargado de implementar la lógica administradora de la transacción a través del protocolo WS-ATOMIC TRANSACTION.

Finalmente los diagramas de secuencia, de clases y el pseudocódigo de los elementos participantes nos brindan una idea amplia de la interacción entre los elementos participantes, cuando estos llevan a cabo el protocolo transaccional durable de dos fases, además de un panorama amplio que complementa la implementación de estos elementos.

EXPOSICIÓN DE COM+ EN ACTIVIDADES TRANSACCIONALES MEDIANTE SERVICIOS WEB

1.1. Introducción

La infraestructura de una aplicación en un ambiente multiusuario esta compuesta por servicios que habilitan a varios usuarios para acceder a la aplicación y su data relacionada al mismo tiempo. Estos servicios mantienen la integridad de la data y los procesos que protegen el orden de trabajo de la aplicación de negocios, la infraestructura de la aplicación incluye: hilos, conexiones a base de datos seguridad y transacciones. Los desarrolladores deben gastar gran parte de su tiempo en crear esta infraestructura bajo la cual estos aplicativos puedan ejecutarse. Extraído de la referencia bibliográfica 10 , revisar apéndice D.

Los servicios de COM+ proveen esta infraestructura, así los desarrolladores pueden concentrarse en construir aplicaciones y no tienen que preocuparse en la infraestructura de soporte. Algunos servicios incluyen soporte para transacciones, manejo de recursos, sincronización y seguridad, COM + provee un procesamiento basado en componentes transaccionales para desarrollar, desplegar y administrar aplicaciones de servidor. Extraído de la referencia bibliográfica 10 , revisar apéndice D.

En la opinión del autor : El trabajo realizado en una aplicación multiusuario distribuida , puede envolver cambios en recursos que son bases de datos y en recursos que no son bases de datos , entidades como colas , a menudo más de un objeto en actividad esta iniciando cambios en los recursos , porque el trabajo en un simple objeto puede fallar , pero tener éxito en otros . El trabajo no puede ser dejado en un estado inconsistente, donde parte del trabajo esta hecho correctamente y otra parte no.

Las transacciones proveen un modelo simple todo o nada para manejar el trabajo. O todos los objetos tienen éxito y su trabajo es perpetrado, o uno o más objetos fallan y su trabajo no es perpetrado. Basándose en el resultado las tablas de la base de datos o los archivos afectados por el trabajo pueden ser cambiados o no como un todo. Ellos no serán dejados en un estado inconsistente.

La programación requerida para manejar las transacciones a través de múltiples entidades bases de datos y no bases de datos es muy problemática para implementar en una aplicación. COM+ provee este servicio de manejo de transacciones como parte de la infraestructura de su aplicación. COM+ crea automáticamente transacciones para los componentes cuando estos son activados. COM+ maneja automáticamente la limpieza y la operación de dejar sin efecto los cambios en una transacción, no se tiene que escribir código

transaccional en el código de sus componentes. COM+ provee un marco de trabajo denominado “Compensando el manejo de recursos”, el cual permite que los recursos que no poseen manejo de transacciones participen en una transacción, esto permite crear componentes que reviertan los cambios en los recursos que manejan bajo una transacción. Extraído de la referencia bibliográfica 10, revisar apéndice D.

1.2. Problema

Cuando COM+ trabaja en actividades transaccionales bajo la misma plataforma - Windows - DTC Data transaction Coordinator, los contextos transaccionales fluyen a través de todos los componentes inmersos, esto lo maneja automáticamente el DTC, los componentes inmersos para propagar sus estados solo tienen que activar o desactivar flags en el contexto transaccional, para que sus votos en la operación transaccional de dos fases tengan efecto a nivel de toda la actividad transaccional. Extraído de la referencia bibliográfica 10 , revisar apéndice D.

En la opinión del autor : El principal problema de este manejo respecto a su exposición en actividades transaccionales multiplataforma, es que no se puede acceder al contexto transaccional para administrar la actividad

transaccional si no se es un componente transaccional, por lo tanto no se puede tener acceso al contexto transaccional si no se trabaja bajo el modelo de desarrollo COM+, y no se puede administrar directamente la actividad transaccional COM+ desde otras plataforma puesto a que el DTC automáticamente maneja esta infraestructura y no la expone granularmente para poder realizar este manejo .

1.3. Solución

Para exponer los componentes COM+ en actividades transaccionales multiplataforma basadas en servicios web proponemos dos acercamientos, que detallamos a continuación:

1.3.1. Contexto de la solución

1.3.1.1. Exposición ACID completa al protocolo durable de dos fases

Como hemos visto el la narración de el problema, cualquier código que ejecute un componente transaccional es enlistado y manejado por el DTC, lo cual imposibilita la exposición directa por parte de un componente Transaccional COM +.

En esta primera versión de la solución proponemos la exposición de los componentes COM + a través del control manual de la transacción por parte de una clase cliente , aplicativo participante , que necesariamente tiene que ser implementada en la plataforma

Windows , esta clase utiliza el patrón muy conocido de fachada , expondrá la funcionalidad de solo un método transaccional perteneciente a un componente COM+ , esta labor se realizará internamente la clase de fachada a través de una clase COM muy poco conocida y utilizada TransactionContext esta clase se encuentra disponible a través del espacio de nombres COMSVCSLib

La clase TransactionContext posee tres metodos que pueden ser utilizados : CreateInstance, Abort, y Commit.

CreateInstance: es utilizado para crear instancias de componentes transaccionales COM+, si el componente está configurado para soportar o requerir transacciones, COM + automáticamente la enlista en la transacción que maneja la clase TransactionContext.

Commit : Cuando el trabajo está completo y no se han presentado exepciones se puede utilizar este método para commitir la transacción , esto termina la transacción , COM + evalúa la transacción , si todos los componentes votaron por comitir la transacción, los cambios son comitidos, si algún componente vota por abortar la transacción todos los cambios adicionales son deshechos .

Abort : si se detecta algún error este método deshace los cambios realizados en la transacción .

El siguiente código de Ejemplo muestra cómo utilizar la clase TransactionContext :

```
Imports COMSVCSLib
Public Class RootClient
   Public Sub DoWork()
      Dim tc As New ITransactionContext
      Dim txComp1 As MyTxClass()
      Dim txComp2 As MyTxClass()
      Try
         txComp1 = tc.CreateInstance("namespace.MyTxClass")
         txComp2 = tc.CreateInstance("namespace.MyTxClass")
         txComp1.DoSubWork()
         txComp2.DoSubWork()
         tc.Commit()
      Catch e As Exception
         tc.Abort()
      End Try
   End Sub
End Class
```

Para utilizar el espacio de nombres COMSVCSLib se debe hacer su importación en el IDE de visual Studio .NET .

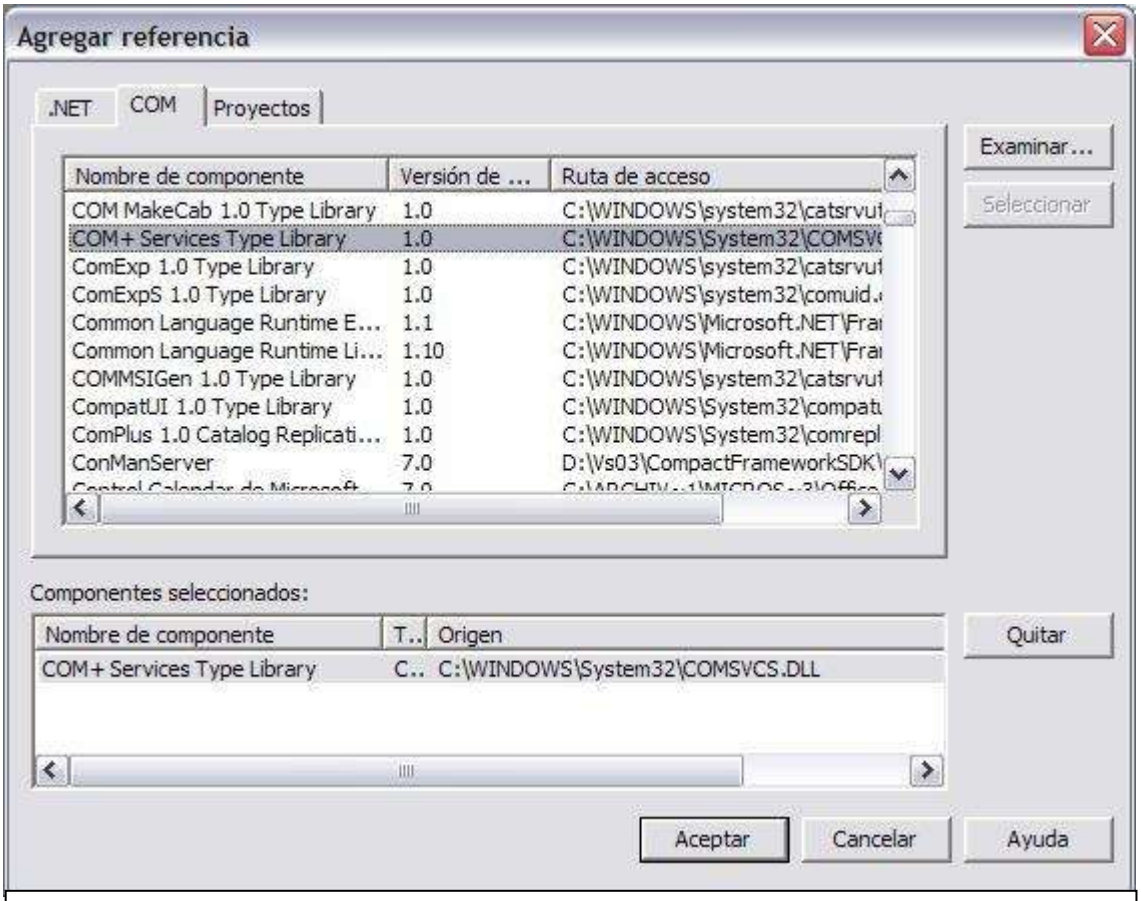


Figura 4: Referencias del proyecto – Fuente: Elaboración Propia

La clase que sirve de fachada deberá implementar los tres métodos que hacen posible la participación de la clase en el protocolo durable de dos fases: Prepare, Rollback , Commit.[40]

Los métodos de la clase de fachada, con más detalle, serían los siguientes:

- **Constructor**

[40] Para más información remitirse al documento WS-Atomic Transaction en el Anexo A.

En este método se debe instanciar la clase TransactionContext de forma completa parte en el stack y parte en heap – se utiliza la palabra reservada New , además de instanciar variables de referencia de los omponentes transaccionales expuestos o a utilizarse, una referencia de código la tenemos a continuación:

```
Dim tc As New ITransactionContext
Dim txComp1 As MyTxClass()
Dim txComp2 As MyTxClass()
```

- **Prepare**

En esta operación el componente de fachada cargará las variables de referencia creadas en el método constructor, con instancias propiamente creadas de los componentes a exponer, acto seguido enviará a ejecutar el método transaccional del o los componentes COM+ expuesto , por medio de el siguiente código :

```
txComp1 =
c.CreateInstance("namespace.MyTxClass")
txComp2 =
c.CreateInstance("namespace.MyTxClass")
txComp1.DoSubWork()
txComp2.DoSubWork()
```

- **Commit**

En esta operación el componente ejecutará el código que le indica al Data transaction Cordinator plasmar los cambios de forma permanente, esto lo hacemos mediante el siguiente código:

```
tc.Commit()
```

- **Rollback**

  Si ocurre algún problema inesperado , por ejemplo si el Componente de fachada genera una excepción interna respecto a las operaciones Prepare o Commit , internamente el componente de fachada debería llamar al código que causa el abort por parte de data transaction cordinator :

  ```
  tc.Abort()
  ```

  Este código también puede ser llamado en cualquier momento por los coordinadores transaccionales Principales o los coordinadores transaccionales Participantes.

  De esta manera la exposición cumpliría con todas las propiedades ACID[41] para los modelos transaccionales.

Cabe resaltar que por medio de esta primera propuesta de exposición de com+ la clase de fachada necesita ser almacenada de

forma permanente hasta el final del protocolo transaccional durable de dos fases[42], como la exposición de esta clase de fachada hacia los elementos que participan en una actividad transaccional multiplataforma es a través de servicios web, estos servicios web pueden aprovechar los métodos de almacenamiento siguientes:

- **Variables de Aplicación en memoria de proceso**

  Las variables de aplicación en los servicios web implementados mediante el marco de desarrollo ASP.NET , son variables accesibles por todas las sesiones o peticiones al servicios , este es un carácter muy importante , porque el flujo de actividades transaccionales propuesto no es síncrono , la implementación de los servicios web recae fuertemente en el protocolo HTTP , en este protocolo no se almacena estados entre peticiones , lo que es lo mismo la clase de fachada sería destruida entre peticiones al servicio web .

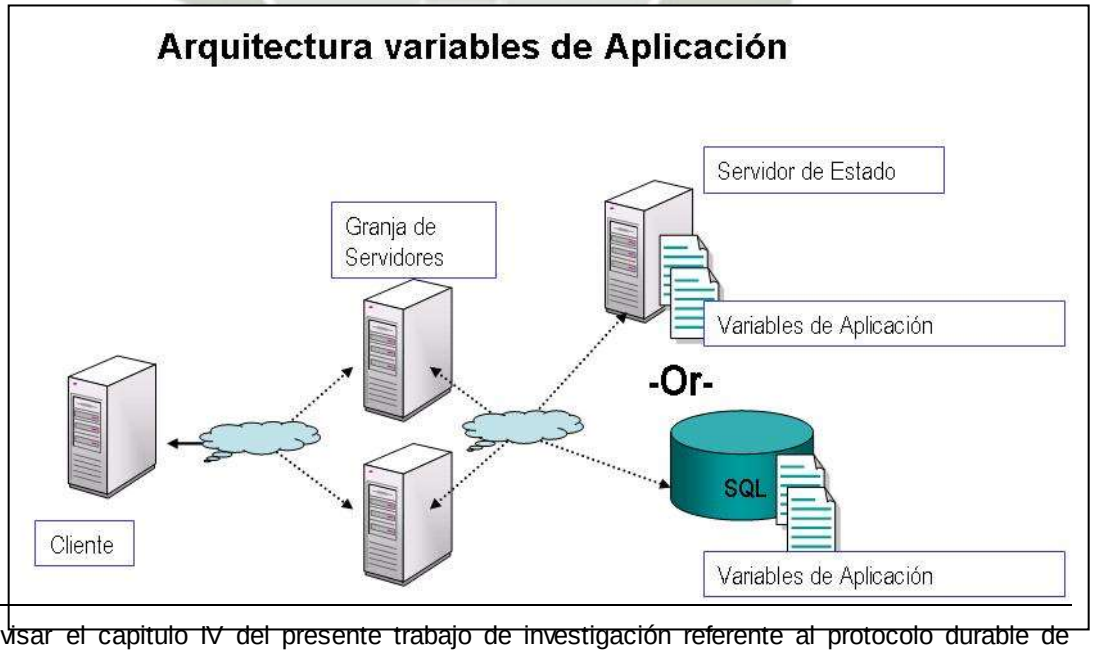
  Uno de los beneficios de este modo de almacenamiento es que las variables estarán en memoria, lo cual nos da una performance muy alta, esto también representa un problema dependiendo de la cantidad de carga, memoria, número peticiones que maneje el servidor que soporte el servicio web que representa al Aplicativo Participante.

[41] Ver el glosario donde se encuentran las propiedades ACID

- **Variables de aplicación serializadas a base de datos**

  En este caso se puede configurar al servicio web que representa al aplicativo participante para que almacene su variables de aplicación ya no en memoria, más bien por medio de una serialización las almacene en una base de datos, esto representa un punto hacia el desarrollo de un aplicativo participante altamente escalable , puesto a que el manejo de esta forma de las variables de aplicación permitía un despliegue del aplicativo en una granja de servidores .

- A continuación presentamos un **esquema de la arquitectura** utilizada por esta parte de la propuesta :



[42] Revisar el capítulo IV del presente trabajo de investigación referente al protocolo durable de dos fases.

Figura 5: Arquitectura Variables de Aplicación. Fuente: Elaboración propia.

- **Un método de almacenamiento alternativo a desarrollar**

  Se pueden desarrollar otros métodos para el almacenamiento permanente de la clase de fachada , por ejemplo serialización a disco , cookies al cliente, etc , cada una debe evaluarse respecto a sus beneficios y restricciones , respeto a la performance , escalabilidad , seguridad que se desea lograr en el aplicativo participante .

1.3.1.2. Aspectos a tener en cuenta respecto a las características transaccionales de los componentes COM+

Para que los componentes COM + puedan trabajar adecuadamente bajo esta forma de exposición en una actividad transaccional multiplataforma hay que tener en cuenta las siguientes consideraciones:

a) La configuración de transacciones a nivel de componente

Esta configuración puede llevar los siguientes valores (Extraídos de la referencia bibliográfica 10 , revisar en anexo C):

- Disabled : Si el componente es llamado bajo una activiadad transaccional COM+, este ignora la actividad transaccional.

- Not Supported (default value) : se le crea un propio contexto de trabajo cuando el ente que llama trabaja bajo transacciones

  Estas dos primeras opciones no deberían ser utilizadas en los componentes expuestos para trabajar en actividades transaccionales multiplataforma

- Supported : trabaja bajo el contexto transaccional COM+ del ente que llama operaciones sobre el componente , es una opción adecuada de exposición puesto a que el contexto COM+ inicial que fluiria sería el de la clase de fachada .

- Required : igual que el atributo Supported , la diferencia está que si el ente que llama no soporta transacciones se crea un contexto transaccional COM+ para que un componente con este atributo pueda ejecutarse ,este no es nuestro caso , pero el atributo es aceptado.

- Requires New : siempre que se llama al componente se crea un nuevo contexto transaccional COM+ el DTC maneja la comunicación entre los contextos transaccionales del ente que llama y del componente llamado , es una opción aceptable , pero por motivos de performance y evitar la sobrecarga al DTc de tener

que crear un nuevo contexto transaccional COM+ se prefiere el atributo Supported .

El código de declaración del componente expuesto quedaría como sigue:

```
<Transaction(TransactionOption.Required)> _
Public Class MyVBClass
Inherits ServicedComponent
Public Sub Method1()
'Perform Work
End Sub
End Class
```

Además de ello este atributo se puede cambiar mediante la herramienta de administración de los aplicativos COM +:

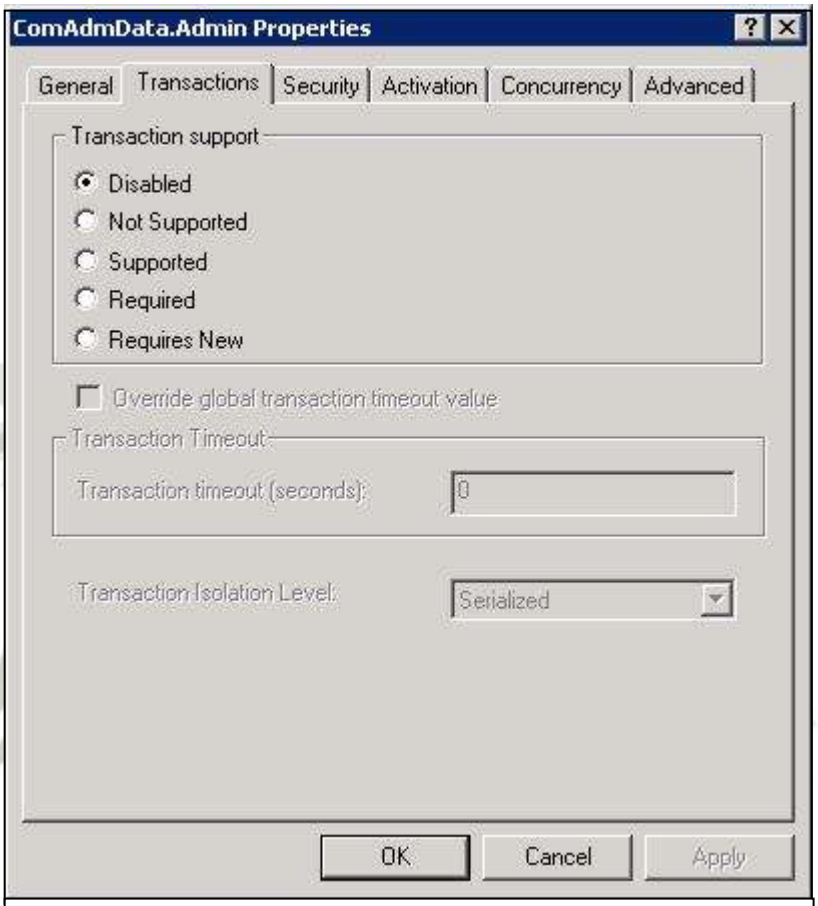


Figura 6: Tipo de Transacción que acepta el componente expuesto – Fuente: Elaboración Propia

b) El bit Listo

Es un bit que es mantenido en cada contexto COM+ su valor por defecto es falso, cuando una operación transaccional COM+ termina y este bit se encuentra desactivado, el componente COM+ se queda activo en memoria preservando su estado, para actividades

transaccionales multiplataforma este comportamiento no es deseado puesto a que generalmente el número de transacciones manejadas por los aplicativos participantes suele ser alto, y cualquier estado que se necesite almacenar debería ser almacenado de forma permanente con otros acercamientos. Se recomienda por lo tanto colocar este bit en true luego de terminar cualquier operación transaccional al interior de un componente COM+.

Se puede lograr este comportamiento por medio e la manipulación del atributo que representa al bit listo dentro del contexto transaccional, mediante código o se puede configurar al componente para que se desactive cada vez que termine de realizar el proceso de un método expuesto, para lograr este comportamiento disponemos de la consola de administración de los aplicativos COM+, aquí hay que tener activo, el check de just in time activation (Extraído de la referencia bibliográfica 10 , revisar en anexo C) :

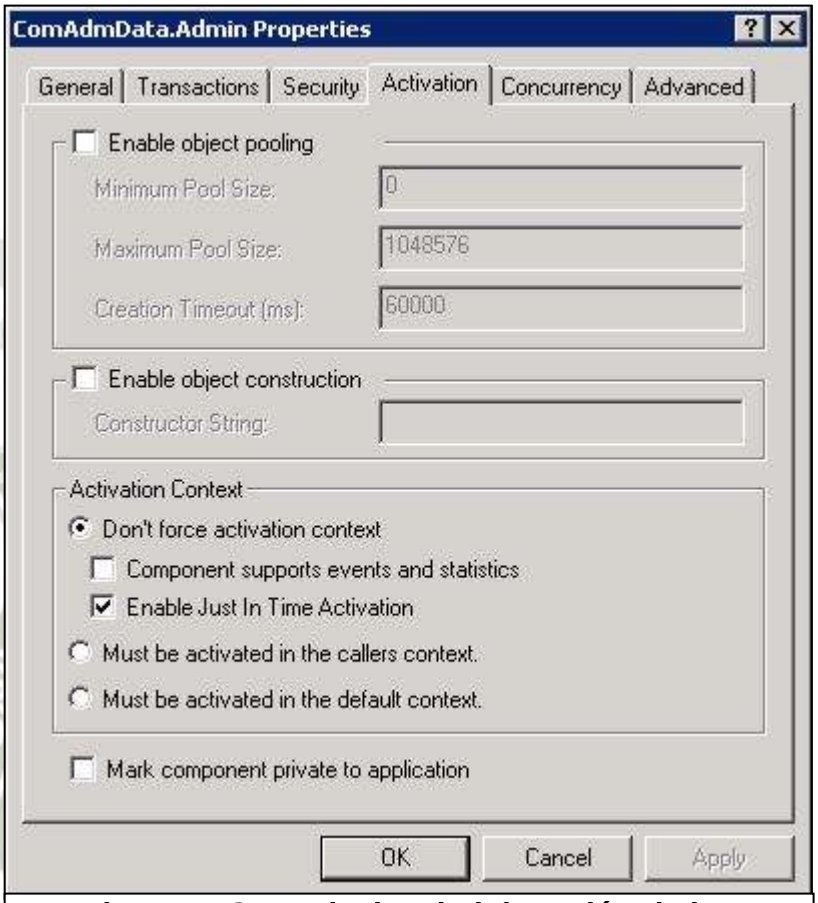


Figura 7: Consola de administración de los aplicativos COM+. Fuente: Elaboración propia.

Esta opción también puede ser configurada mediante código en la definición del componente COM+ :

```
<JustInTimeActivation(True)> _
Public Class MyVBClass
   Inherits ServicedComponent
   Implements IMyClass
```

```
    Public Sub Method1()

    End Sub

End Class
```

c) Tiempo fuera

Información extraída de referencia bibliográfica 10, revisar el anexo C. El tiempo máximo que se utiliza para administrar un transacción COM+ es uno de los aspectos más importantes para la exposición de componentes COM+ Transaccionales mediante actividades transaccionales multiplataforma.

Hay dos formas de controlar estos tiempos máximos: a nivel de todos los DTC inmersos en la transacción COM+ o a nivel de cada componente, hay que tener en cuenta que en una actividad transaccional multiplataforma hay mensajes que van y vienen entre los elementos participantes, como son los coordinadores transaccionales tanto Principal como Participantes.

El aplicativo iniciador y los aplicativos participantes, la comunicación que se da entre ellos debe ser por medio de red, las redes traen consigo ciertos riesgos de operación, algún nodo de la red sobre el cual se ejecute uno de los elementos transaccionales puede fallar.

Por lo tanto como recomendación el tiempo manejado por el componente COM+ expuesto y los coordinadores inmersos en la transacción COM+ expuesta , deberían ser mayores que el tiempo manejado por la actividad transaccional , lamentablemente estos tipos de configuraciones no se pueden manejar por cada petición tienen que ser colocados con anterioridad , debe tenerse en cuenta además que a mayores tiempos de duración sobre las transacciones COM+ los DTCs manejarán una mayor carga de información respecto a las transacciones que ellos administran.

El tiempo máximo para resolver una transacción a nivel de todo un DTC puede modificarse mediante la consola de administración COM+ propiedades del DTC, el tiempo ingresado se da en base a segundos:

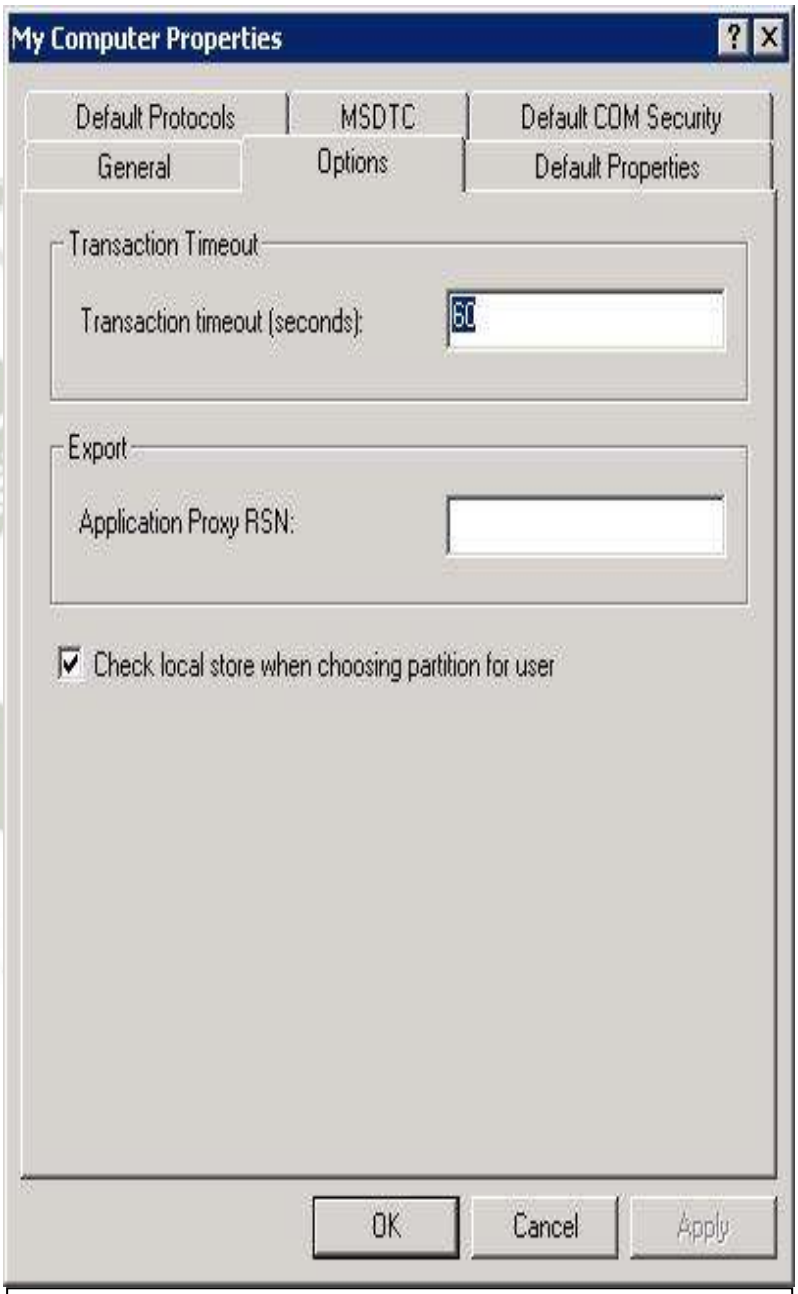


Figura 8: Consola de administración COM+ propiedades del DTC – Fuente: Elaboración Propia

El tiempo máximo controlado a nivel de cada componente se puede modificar vía código, mediante el atributo timeout en la definición del componente:

```
<Transaction(TransactionOption.Required,
Timeout:=120)>
Public Class MyVBClass
Inherits ServicedComponent
Public Sub Method1()
End Sub
End Class
```

Además el tiempo máximo para resolver una transacción puede ser administrado mediante la consola de administración COM+, propiedades del componente:

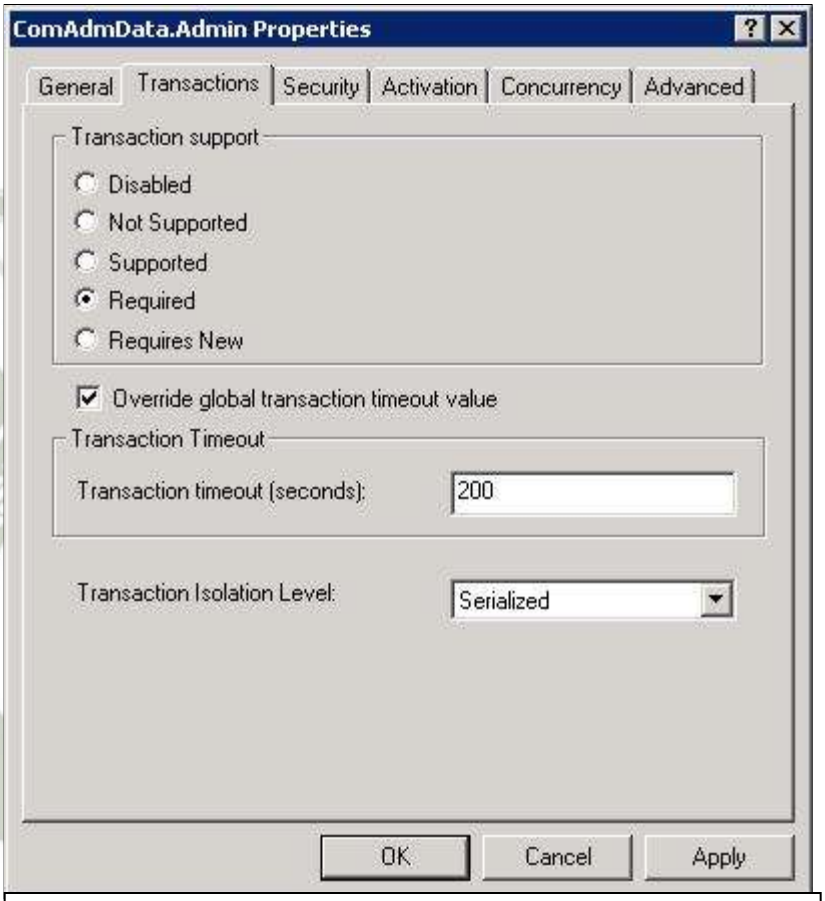


Figura 9: Consola de administración COM+ propiedades del Componente –Fuente: Elaboración Propia

1.3.2. Exposición de Componentes COM+ mediante Operaciones de Compensación

Mediante esta propuesta también tendríamos la clase de fachada con los mismos métodos expuestos para participar en la actividad transaccional multiplataforma: Prepare, Rollback , Commit . Los métodos de la clase de fachada, con más detalle, serían los siguientes:

- **Prepare :**

  En este método se instanciar variables de referencia de los componentes transaccionales expuestos o a utilizarse, una referencia de código la tenemos a continuación no habría operación de preparación propiamente dicha, aquí no se reservarían recursos transaccionales para continuar con la actividad transaccional.

- **Commit**

  En esta operación el componente de fachada cargará las variables de referencia creadas en el método prepare, con instancias propiamente creadas de los componentes a exponer, acto seguido llamará a los métodos transaccionales, el DTC y la infraestructura COM+ se encargarán internamente a la llamada de comitir o abortar la transacción interna a COM+ .

- **Rollback**

  En esta operación el componente de fachada cargará las variables de referencia creadas en el método pepare, con instancias propiamente creadas de los componentes a exponer, acto seguido llamará a los métodos transaccionales que ejecutan acciones inversas o e compensación respecto a los métodos transaccionales invocados en la operación commit.

Este tipo de exposición de las transacciones COM+ Puede dejar la información en un estado inconsistente, lo cual permitiría en otros elementos en otras transacciones, ver elementos fantasmas hacer lecturas sucias, etc.

Esta forma de exposición debe ser evaluada respecto a los objetivos y diseño de los sistemas a desarrollar, teniendo en cuenta además el trabajo extra que significa el análisis y desarrollo de los métodos de compensación, estos métodos de compensación requieren que se les pasen todos los dato necesarios para realizar la operación de compensación y recobrar datos de un método de almacenamiento permanente para completar su operación de compensación.

1.4. Resumen del capítulo

Hemos conocido dos formas de realizar la exposición de los componentes COM+ para que trabajen en actividades transaccionales multiplataforma.

La más confiable de las dos es la exposición ACID completa al protocolo durable de dos fases, otra exposición permisible según el diseño y comportamiento deseado en los sistemas a desarrollar es Exposición de Componentes COM+ mediante Operaciones de Compensación.

Para la exposición ACID completa al protocolo durable de dos fases se tienen que tener en cuenta las configuraciones respecto a las transacciones de cada uno de los componentes COM+ expuestos.

ANEXO B - GLOSARIO

Elementos participantes en una actividad transaccional Multiplataforma

- **Actividades transaccionales multiplataforma todo o nada**

  Es un proceso de software en el cual intervienen varias llamadas a diferentes componentes, si toda la actividad termina de forma satisfactoria los cambios realizados por los componentes que intervienen en la actividad son permanentes, si la actividad no termina de forma satisfactoria, los cambios tienen que ser deshechos por todos los componentes que han participado en la actividad, de allí su característica todo o nada.

- **Aplicativo Iniciador**

  Aplicativo que inicia la actividad transaccional, es el encargado de elegir un coordinador Transaccional Principal

- **Aplicativo Participante**

  Es al cual el Aplicativo Participante le pide funcionalidad de forma transaccional, este aplicativo participan en la actividad transaccional exponiendo una interfaz transaccional, que es un punto al cual pueden fluir mensajes pertenecientes a un protocolo transaccional.

- **Coordinador Transaccional Participante**

  Encargado de Coordinar la actvidad transaccional con un Aplicativo Participante, se encarga de enviar los mensajes transaccionales a la interfaz transaccional del aplicativo participante, también propaga hacia en

coordinador transaccional principal los mensajes transaccionales que genera el aplicativo participante.

- **Coordinador Transaccional Principal**

  Elemento encargado de Coordinar tanto el protocolo de coordinación como los mensajes que del protocolo transaccional que fluyen entre los elementos inmersos en la actividad transaccional multiplataforma.

Propiedades ACID

- **Aislamiento**

  Las transacciones simultáneas se aíslan de las actualizaciones de otras transacciones incompletas. Estas actualizaciones no constituyen un estado coherente. Esta propiedad se denomina con frecuencia serializabilidad. Por ejemplo, una segunda transacción que recorre la lista doblemente vinculada mencionada anteriormente en el ejemplo de coherencia verá la lista antes o después de la inserción, pero sólo verá los cambios completados.

- **Atomicidad**

  Una transacción o bien se confirma o se anula. Si se confirma una transacción, permanecen todos sus efectos. Si se anula, se deshacen todos sus efectos. Por ejemplo, al cambiar el nombre de un objeto se crea el nuevo nombre y se elimina el anterior (confirmación) o no cambia nada (anulación).

- **Coherencia**

  Una transacción es una transformación correcta del estado del sistema. Mantiene las constantes de estado. Por ejemplo, al agregar un elemento a una lista doblemente vinculada, se actualizan los cuatro punteros hacia delante y hacia atrás.

- **Durabilidad**

  Una vez que se confirma una transacción, sus efectos continuarán aunque haya errores en el sistema. Por ejemplo, después del cambio de nombre en el ejemplo de atomicidad, el objeto tendrá el nuevo nombre incluso aunque se produzca un error en el sistema y se reinicie justo después de que se complete la confirmación.

Varios

- **Actividad Transaccional**

  Conjunto de Aplicativos que comparten la misma transacción lógica.

- **COM+**

  Component Object Model Plus, es una plataforma de Microsoft para componentes de software. Esta plataforma es utilizada para permitir la comunicación entre procesos y la creación dinámica de objetos, en cualquier lenguaje de programación que soporte dicha tecnología.

- **SOA**

  La Arquitectura Orientada a Servicios (en inglés Service-oriented architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

- **Web**

  La World Wide Web, la Web o WWW, es un sistema de navegador Web para extraer elementos de información llamados "documentos" o "páginas Web". Puede referirse a "una Web" como una página, sitio o conjunto de sitios que proveen información por los medios descritos, o a "la Web", que es la enorme e interconectada red disponible prácticamente en todos los sitios de Internet.

- **DTC**

  Coordinador de transacciones distribuidas de Microsoft, dirige t coordina las transacciones de un componente COM+.

- **W3C**

  El World Wide Web Consortium, abreviado W3C, es un consorcio internacional que produce estándares para la World Wide Web. Está dirigida por Tim Berners-Lee, el creador original de URL (Uniform Resource Locator, Localizador Uniforme de Recursos), HTTP (HyperText Transfer Protocol, Protocolo de Transferencia de HiperTexto) y HTML (Lenguaje de Marcado

de HiperTexto) que son las principales tecnologías sobre las que se basa la Web.

- **Patrones de Diseño**

Los Patrones de Diseño (Design Patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

Objetivos de los patrones

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.

- Eliminar la creatividad inherente al proceso de diseño.

- **CORBA**

  En computación, CORBA (Common Object Request Broker Architecture — arquitectura común de intermediarios en peticiones a objetos), es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

  CORBA fue definido y está controlado por el Object Management Group (OMG) que define las APIs, el protocolo de comunicaciones y los mecanismos necesarios para permitir la interoperabilidad entre diferentes aplicaciones escritas en diferentes lenguajes y ejecutadas en diferentes plataformas, lo que es fundamental en computación distribuida.

  En un sentido general CORBA "envuelve" el código escrito en otro lenguaje en un paquete que contiene información adicional sobre las capacidades del código que contiene, y sobre cómo llamar a sus métodos. Los objetos que resultan pueden entonces ser invocados desde otro programa (u objeto CORBA) desde la red. En este sentido CORBA se puede considerar como un formato de documentación legible por la máquina, similar a un archivo de cabeceras pero con más información.

  CORBA utiliza un lenguaje de definición de interfaces (IDL) para especificar los interfaces con los servicios que los objetos ofrecerán. CORBA puede

especificar a partir de este IDL la interfaz a un lenguaje determinado, describiendo cómo los tipos de dato CORBA deben ser utilizados en las implementaciones del cliente y del servidor. Implementaciones estándar existen para Ada, C, C++, Smalltalk, Java y Python. Hay también implementaciones para Perl y TCL.

Al compilar una interfaz en IDL se genera código para el cliente y el servidor (el implementador del objeto). El código del cliente sirve para poder realizar las llamadas a métodos remotos. Es el conocido como stub, el cual incluye un proxy (representante) del objeto remoto en el lado del cliente. El código generado para el servidor consiste en unos skeletons (esqueletos) que el desarrollador tiene que rellenar para implementar los métodos del objeto. CORBA es más que una especificación multiplataforma, también define servicios habitualmente necesarios como seguridad y transacciones.

- **RMI**

RMI (Java Remote Method Invocation) es un mecanismo ofrecido en Java para invocar un método remotamente. Al ser RMI parte estándar del entorno de ejecución Java usarlo provee un mecanismo simple en una aplicación distribuida que solamente necesita comunicar servidores codificados para Java. Si se requiere comunicarse con otras tecnologías debe usarse CORBA o SOAP en lugar de RMI.

Al estar específicamente diseñado para Java RMI puede darse el lujo de ser muy amigable para los programadores, proveyendo pasaje por referencia de objetos (cosa que no hace SOAP), "recolección de basura" distribuida y pasaje de tipos arbitrarios (funcionalidad no provista por CORBA).

Por medio de RMI, un programa Java puede exportar un objeto. A partir de esa operación este objeto está disponible en la red, esperando conexiones en un puerto TCP. Un cliente puede entonces conectarse e invocar métodos. La invocación consiste en el "marshaling" de los parámetros (utilizando la funcionalidad de "serialización" que provee Java), luego se sigue con la invocación del método (cosa que sucede en el servidor). Mientras esto sucede el llamador se queda esperando por una respuesta. Una vez que termina la ejecución el valor de retorno (si lo hay) es serializado y enviado al cliente. El código cliente recibe este valor como si la invocación hubiera sido local.

ANEXO C - WEB SERVICES ATOMIC TRANSACTION (WSATOMICTRANSACTION)

November 2004

Authors

Luis Felipe Cabrera, Microsoft

George Copeland, Microsoft

Max Feingold, Microsoft

Tom Freund, IBM

Jim Johnson, Microsoft

Chris Kaler, Microsoft

Johannes Klein, Microsoft

David Langworthy, Microsoft (Editor)

Anthony Nadalin, IBM

David Orchard, BEA Systems

Ian Robinson, IBM

Tony Storey, IBM

Satish Thatte, Microsoft

Copyright Notice

that you include the following on ALL copies of the “Web Services Atomic Transaction” Specification that you make:

1. A link or URL to the “Web Services Atomic Transaction” Specification at one of the Authors’ websites

2. The copyright notice as shown in the “Web Services Atomic Transaction” Specification.

IBM, Microsoft and BEA (collectively, the “Authors”) each agree to grant you a license, under royalty-free and otherwise reasonable, non-discriminatory terms and conditions, to their respective essential patent claims that they deem necessary to implement the “Web Services Atomic Transaction” Specification.

THE “WEB SERVICES ATOMIC TRANSACTION” SPECIFICATION IS PROVIDED "AS IS," AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE “WEB SERVICES ATOMIC TRANSACTION” SPECIFICATION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE “WEB SERVICES ATOMIC TRANSACTION” SPECIFICATION.

The name and trademarks of the Authors may NOT be used in any manner,

including advertising or publicity pertaining to the “Web Services Atomic Transaction” Specification or its contents without specific, written prior permission. Title to copyright in the “Web Services Atomic Transaction” Specification will at all times remain with the Authors. No other rights are granted by implication, estoppel or otherwise.

Abstract

This specification provides the definition of the atomic transaction coordination type that is to be used with the extensible coordination framework described in the WS-Coordination specification. The specification defines three specific agreement coordination protocols for the atomic transaction coordination type: completion, volatile two-phase commit, and durable two-phase commit. Developers can use any or all of these protocols when building applications that require consistent agreement on the outcome of short-lived distributed activities that have the all-or-nothing property.

Composable Architecture

By using the SOAP [SOAP]and WSDL [WSDL] extensibility model, SOAP-based and WSDL-based specifications are designed to work together to define a rich Web services environment. As such, WS-AtomicTransaction by itself does not define all features required for a complete solution. WS-AtomicTransaction is a building block used with other specifications of Web services (e.g., WS-Coordination, WS-Security) and application-specific protocols that are able to accommodate a wide variety of coordination protocols

related to the coordination actions of distributed applications.

Status

WS-AtomicTransaction and related specifications are provided for use as-is and for review and evaluation only. Microsoft, BEA, and IBM will solicit your contributions and suggestions in the near future. Microsoft, BEA, and IBM make no warrantees or representations regarding the specification in any manner whatsoever.

Acknowledgments

The following individuals have provided invaluable input into the design of the WSAtomicTransaction specification:

Francisco Curbera, IBM

Sanjay Dalal, BEA Systems

Doug Davis, IBM

Gert Drapers, Microsoft

Don Ferguson, IBM

Kirill Garvylyuk, Microsoft

Frank Leymann, IBM

Thomas Mikalsen, IBM

Jagan Peri, Microsoft

John Shewchuk, Microsoft

Alex Somogyi, BEA Systems

Stefan Tai, IBM

Sanjiva Weerawarana, IBM

Preconditions

We also wish to thank the technical writers and development reviewers who provided feedback to improve the readability of the specification.

Table of Contents

1. Introduction

The current set of Web service specifications [WSDL] [SOAP] defines protocols for Web service interoperability. Web services increasingly tie together a number of participants forming large distributed applications. The resulting activities may have complex structure and relationships.

The WS-Coordination specification defines an extensible framework for defining coordination types. This specification provides the definition of an atomic transaction coordination type used to coordinate activities having an "all or nothing" property.

Atomic transactions commonly require a high level of trust between participants and are short in duration. The Atomic Transaction specification defines protocols that enable existing transaction processing systems to wrap their proprietary protocols and interoperate across different hardware and software vendors.

To understand the protocol described in this specification, the following assumptions are made:

• The reader is familiar with existing standards for two-phase commit protocols and with commercially available implementations of such protocols. Therefore this section includes only those details that are essential to understanding the

protocols described.

• The reader is familiar with the WS-Coordination [WSCOOR] specification that defines the framework for the WS-AtomicTransaction coordination protocols.

• The reader is familiar with WS-Addressing [WSADDR] and WS-Policy [WSPOLICY].

Atomic transactions have an all-or-nothing property. The actions taken prior to commit are only tentative (i.e., not persistent and not visible to other activities). When an application finishes, it requests the coordinator to determine the outcome for the transaction. The coordinator determines if there were any processing failures by asking the participants to vote. If the participants all vote that they were able to execute successfully, the coordinator commits all actions taken. If a participant votes that it needs to abort or a participant does not respond at all, the coordinator aborts all actions taken. Commit makes the tentative actions visible to other transactions. Abort makes the tentative actions appear as if the actions never happened. Atomic transactions have proven to be extremely valuable for many applications. They provide consistent failure and recovery semantics, so the applications no longer need to deal with the mechanics of determining a mutually agreed outcome decision or to figure out how to recover from a large number of possible inconsistent states.

Atomic Transaction defines protocols that govern the outcome of atomic transactions. It is expected that existing transaction processing systems wrap their proprietary mechanisms and interoperate across different vendor implementations.

1.1 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [KEYWORDS]. Namespace URIs of the general form "some-URI" represents some applicationdependent or context-dependent URI as defined in RFC2396 [URI].

1.2 Namespace

The XML namespace [XML-ns] URI that MUST be used by implementations of this

specification is:

http://schemas.xmlsoap.org/ws/2004/10/wsat

This is also used as the CoordinationContext type for atomic transactions.

The following namespaces are used in this document:

Prefix Namespace

S http://www.w3.org/2003/05/soap-envelope

wscoor http://schemas.xmlsoap.org/ws/2004/10/wscoor

wsat http://schemas.xmlsoap.org/ws/2004/10/wsat

If an action URI is used then the action URI MUST consist of the wsat namespace URI concatenated with the "/" character and the element name. For example:

http://schemas.xmlsoap.org/ws/2004/10/wsat/Commit

1.3 XSD and WSDL Files

The following links hold the XML schema and the WSDL declarations defined in this document.

http://schemas.xmlsoap.org/ws/2004/10/wsat/wsat.xsd

http://schemas.xmlsoap.org/ws/2004/10/wsat/wsat.wsdl

Soap bindings for the WSDL documents defined in this specification MUST use "document" for the *style* attribute.

2. Atomic Transaction Context

Atomic Transaction builds on WS-Coordination, which defines an activation and a registration service. Example message flows and a complete description of creating and registering for coordinated activities is found in the WS-Coordination

specification [WSCOOR].

The Atomic Transaction coordination context must flow on all application messages involved with the transaction.

Atomic Transaction adds the following semantics to the CreateCoordinationContext

operation on the activation service.

• If the request includes the CurrentContext element, the target coordinator is interposed as a subordinate to the coordinator stipulated inside the CurrentContext element.

• If the request does not include a CurrentContext element, the target coordinator creates a new transaction and acts as the root.

A coordination context may have an Expires attribute. This attribute specifies the earliest point in time at which a transaction may be terminated solely due to its length of operation. From that point forward, the transaction manager may elect to unilaterally roll back the transaction, so long as it has not transmitted a Commit or a Prepared notification.

The Atomic Transaction protocol is identified by the following coordination type:

http://schemas.xmlsoap.org/ws/2004/10/wsat

3. Atomic Transaction Protocols

This specification defines the following protocols for atomic transactions.

• **Completion:** The completion protocol initiates commitment processing. Based on each protocol's registered participants, the coordinator begins with Volatile 2PC then proceeds through Durable 2PC. The final result is signaled to the initiator.

• **Two-Phase Commit (2PC)**: The 2PC protocol coordinates registered participants to reach a commit or abort decision, and ensures that all participants are informed of the final result. The 2PC protocol has two variants:

• **Volatile 2PC:** Participants managing volatile resources such as a cache should register for this protocol.

• **Durable 2PC:** Participants managing durable resources such as a database should register for this protocol.

A participant can register for more than one of these protocols by sending multiple Register messages.

3.1 Preconditions

The correct operation of the protocols requires that a number of preconditions MUST be established prior to the processing:

1. The source MUST have knowledge of the destination's policies, if any, and the source MUST be capable of formulating messages that adhere to this policy.

2. If a secure exchange of messages is required, then the source and destination MUST have a security context.

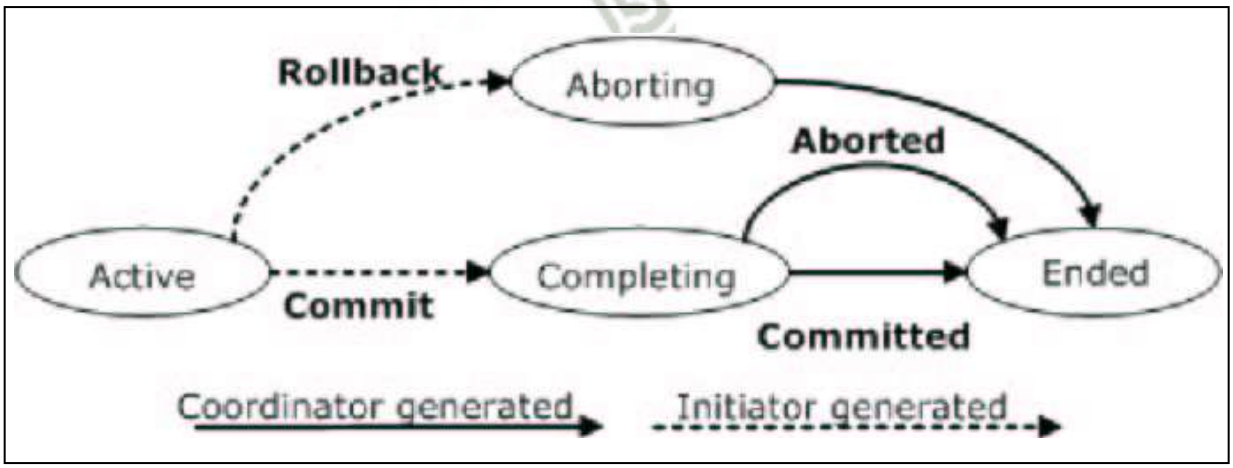
3.2 Completion Protocol

The Completion protocol is used by an application to tell the coordinator to either try to commit or abort an atomic transaction. After the transaction has completed, a status is returned to the application.

An initiator registers for this protocol using the following protocol identifier:

http://schemas.xmlsoap.org/ws/2004/10/wsat/Completion

The diagram below illustrates the protocol abstractly:



The coordinator accepts:

Commit

Upon receipt of this notification, the coordinator knows that the participant has completed application processing and that it should attempt to commit the transaction.

Rollback

Upon receipt of this notification, the coordinator knows that the participant has terminated application processing and that it should abort the transaction.

The initiator accepts:

Committed

Upon receipt of this notification, the initiator knows that the coordinator reached a decision to commit.

Aborted

Upon receipt of this notification, the initiator knows that the coordinator reached a decision to abort.

Conforming implementations must implement Completion.

3.3 Two-Phase Commit Protocol

The Two-Phase Commit (2PC) protocol is a Coordination protocol that defines how multiple participants reach agreement on the outcome of an atomic transaction. The 2PC protocol has two variants: Durable 2PC and Volatile 2PC.

3.3.1 Volatile Two-Phase Commit Protocol

Upon receiving a Commit notification in the completion protocol, the root coordinator begins the prepare phase of all participants registered for the Volatile 2PC protocol.

All participants registered for this protocol must respond before a Prepare is issued to a participant registered for Durable 2PC. Further participants may register with the coordinator until the coordinator issues a Prepare to any durable participant. A volatile recipient is not guaranteed to receive a notification of the transaction's outcome.

Participants register for this protocol using the following protocol identifier:

http://schemas.xmlsoap.org/ws/2004/10/wsat/Volatile2PC

3.3.2 Durable Two-Phase Commit Protocol

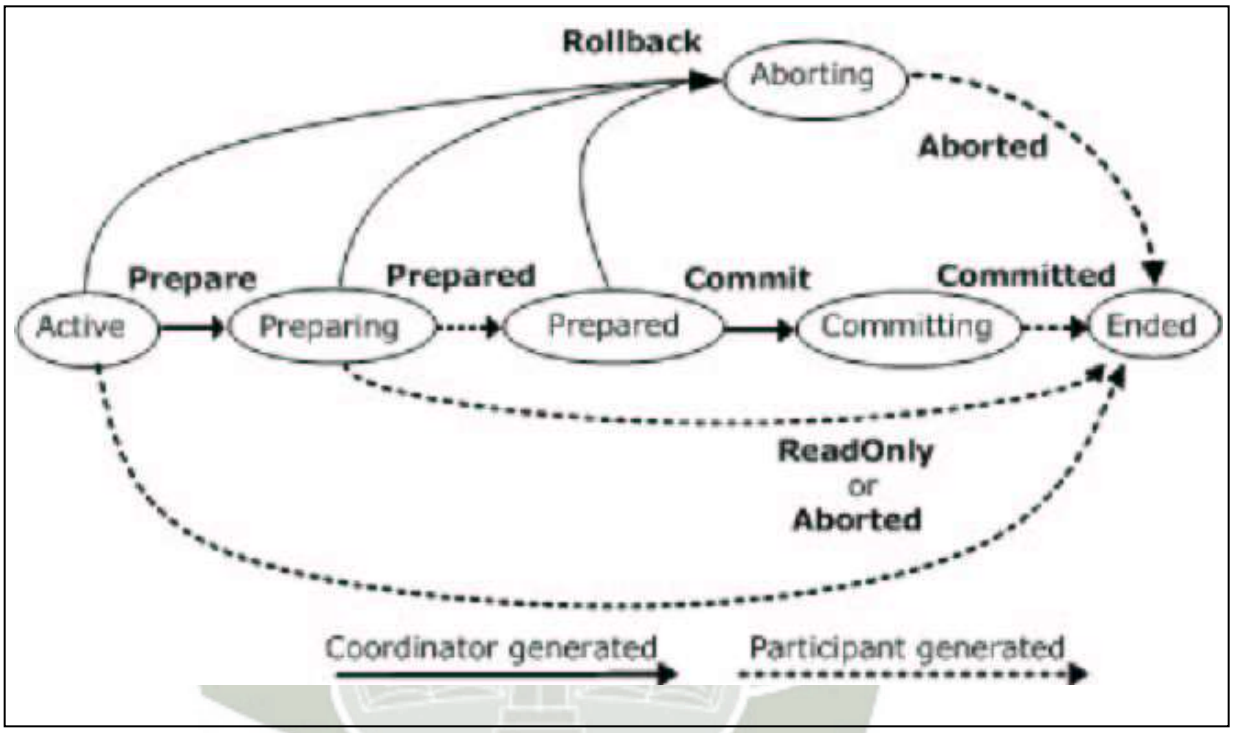
After receiving a Commit notification in the completion protocol and upon successfully completing the prepare phase for Volatile 2PC participants, the root coordinator begins the Prepare phase for Durable 2PC participants. All participants registered for this protocol must respond Prepared or ReadOnly before a Commit notification is issued to a participant registered for either protocol. Participants register for this protocol using the following protocol

identifier:

http://schemas.xmlsoap.org/ws/2004/10/wsat/Durable2PC

3.3.3 2PC Diagram and Notifications

The diagram below illustrates the protocol abstractly:



The participant accepts:

Prepare

Upon receipt of this notification, the participant knows to enter phase 1 and vote on the outcome of the transaction. If the participant does not know of the transaction, it must vote to abort. If the participant has already voted, it should resend the same vote.

Rollback

Upon receipt of this notification, the participant knows to abort, and forget, the transaction. This notification can be sent in either phase 1 or phase 2. Once sent, the coordinator may forget all knowledge of this transaction.

Commit

Upon receipt of this notification, the participant knows to commit the transaction. This notification can only be sent after phase 1 and if the participant voted to commit. If the participant does not know of the transaction, it must send a Committed notification to the coordinator.

The coordinator accepts:

Prepared

Upon receipt of this notification, the coordinator knows the participant is prepared and votes to commit the transaction.

ReadOnly

Upon receipt of this notification, the coordinator knows the participant votes to commit the transaction, and has forgotten the transaction. The participant does not wish to participate in phase 2.

Aborted

Upon receipt of this notification, the coordinator knows the participant has

aborted, and forgotten, the transaction.

Committed

Upon receipt of this notification, the coordinator knows the participant has committed the transaction. That participant may be safely forgotten.

Replay

Upon receipt of this notification, the coordinator may assume the participant has suffered a recoverable failure. It should resend the last appropriate protocol notification.

Conforming implementations MUST implement the 2PC protocol.

4. Policy Assertions

WS-Policy [WSPOLICY], WS-PolicyAttachment [WSPOLICYATTACH], and WSPolicyAssertions [WSPOLICYASSERT] collectively define a framework, model, and grammar for expressing the capabilities, requirements, and general characteristics of entities in an XML Web services-based system. This specification leverages the WSPolicy family of specifications to enable participants and coordinators to describe and advertise their capabilities and/or requirements. The set of policy assertions for atomic transaction is defined below.

4.1. Spec Version

The protocol determines invariants maintained by the reliable messaging endpoints and the directives used to track and manage the delivery of messages. The assertion that will be used to identify the protocol (and version) either used or supported (depending on context) is the wsp:SpecVersion assertion that is defined in the WS-PolicyAssertions specification [WSPOLICYASSERT].

An example use of this assertion to indicate an endpoint's support for the atomic transaction protocol follows:

<wsp:SpecVersion

wsp:URI="http://schemas.xmlsoap.org/ws/2004/10/wsat"

wsp:Usage="wsp:Required"/>

4.2 Protocols

This section establishes well-known names for the protocols supported by atomic transactions.

The following pseudo schema defines these elements:

<wsat:Completion ... />

<wsat:DurableTwoPhaseCommit ... />

<wsat:VolatileTwoPhaseCommit ... />

The following describes the attributes and tags listed in the syntax above:

/wsat:Completion

This element is a policy assertion as defined in WS-Policy Assertions. It indicates support for the Completion Protocol (Section 3.2)

/wsat:DurableTwoPhaseCommit

This element is a policy assertion as defined in WS-Policy Assertions. It indicates support for the Two Phase Commit Protocol (Section 3.3.2)

/wsat:VolatileTwoPhaseCommit

This element is a policy assertion as defined in WS-Policy Assertions. It indicates support for the Two Phase Commit Protocol (Section 3.3.1)

5. Transaction Faults

In addition to the faults defined in WS-Coordination, the following faults are available to Atomic Transaction implementations.

5.1 InconsistentInternalState

This fault is sent by a participant to indicate that it cannot fulfill its obligations. This indicates a global consistency failure and is an unrecoverable condition. The qualified name of the fault code is:

wsat:InconsistentInternalState

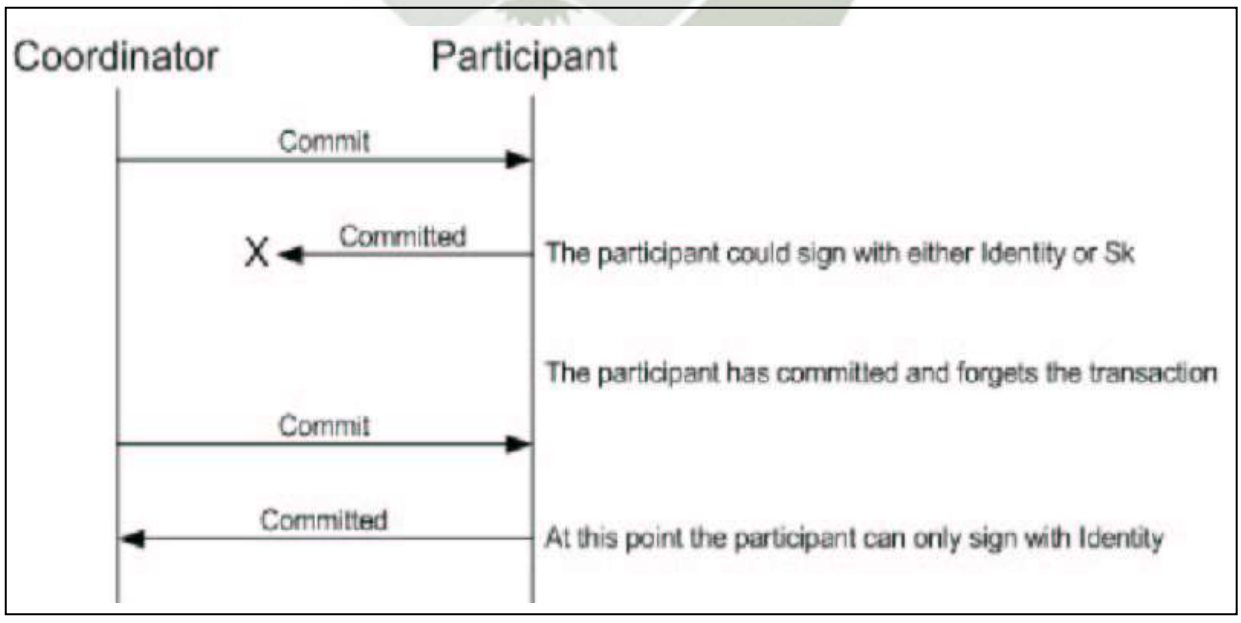
6. Security Model

The security model for atomic transactions builds on the model defined in WSCoordination [WSCOOR]. That is, services have policies specifying their requirements and requestors provide claims (either implicit or explicit) and the requisite proof of those claims. Coordination context creation establishes a base secret which can be delegated by the creator as appropriate.

Because atomic transactions represent a specific use case rather than the

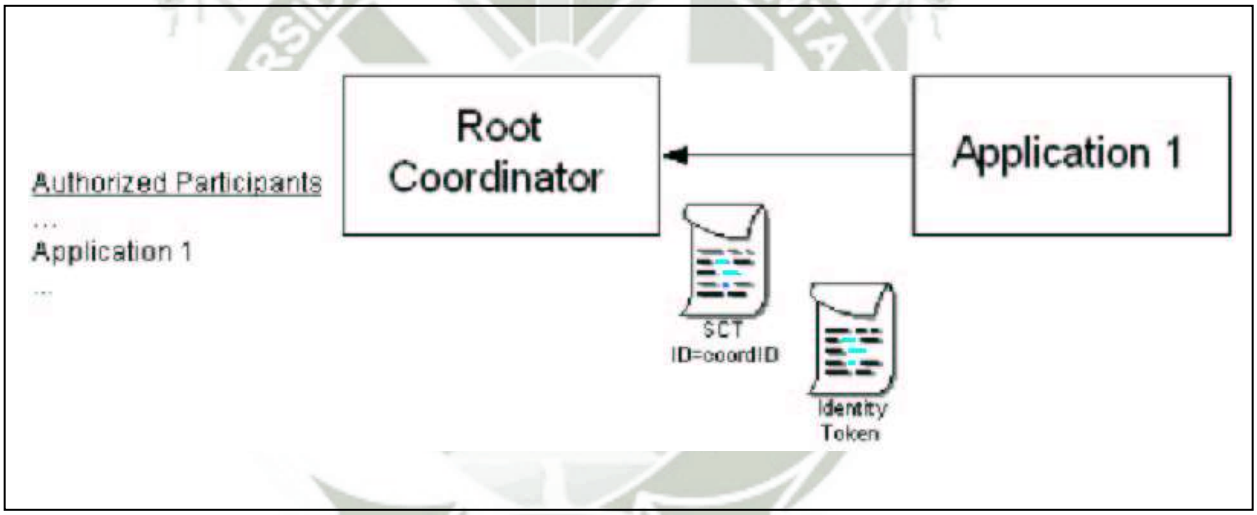
general nature of coordination contexts, additional aspects of the security model can be pecified. All access to atomic transaction protocol instances is on the basis of identity. The nature of transactions, specifically the uncertainty of systems means that the security context established to register for the protocol instance may not be available for the entire duration of the protocol. Consider for example the scenarios where a participant has committed its part of the transaction, but for some reason the coordinator never receives acknowledgement of the commit. The result is that when communication is re-established in the future, the coordinator will attempt to confirm the commit status of the participant, but the participant, having committed the transaction and forgotten all information associated with it, no longer has access to the special keys associated with the token.

The participant can only prove its identity to the coordinator when it indicates that the specified transaction is not in its log and assumed committed. This is illustrated in the figure below:



There are, of course, techniques to mitigate this situation but such options will not always be successful. Consequently, when dealing with atomic transactions, it is critical that identity claims always be proven to ensure that correct access control is maintained by coordinators.

There is still value in coordination context-specific tokens because they offer a bootstrap mechanism so that all participants need not be pre-authorized. As well, it provides additional security because only those instances of an identity with access to the token will be able to securely interact with the coordinator (limiting privileges strategy). This is illustrated in the figure below:



The "list" of authorized participants ensures that application messages having a coordination context are properly authorized since altering the coordination context ID will not provide additional access unless (1) the bootstrap key is provided, or (2) the requestor is on the authorized participant "list" of identities.

7. Security Considerations

It is strongly RECOMMENDED that the communication between services be secured using the mechanisms described in WS-Security [WSSec]. In order to properly secure messages, the body and all relevant headers need to be included in the signature. Specifically, the <wscoor:CoordinationContext> header needs to be signed with the body and other key message headers in order to "bind" the two

together.

In the event that a participant communicates frequently with a coordinator, it is RECOMMENDED that a security context be established using the mechanisms described in WS-Trust [WSTrust] and WS-SecureConversation [WSSecConv] allowing for potentially more efficient means of authentication.

It is common for communication with coordinators to exchange multiple messages. As a result, the usage profile is such that it is susceptible to key attacks. For this reason it is strongly RECOMMENDED that the keys be changed frequently. This "rekeying" can be effected a number of ways. The following list outlines four common techniques:

• Attaching a nonce to each message and using it in a derived key function with the shared secret.

• Using a derived key sequence and switch "generations"

• Closing and re-establishing a security context (not possible for delegated keys)

• Exchanging new secrets between the parties (not possible for delegated keys)

It should be noted that the mechanisms listed above are independent of the SCT and secret returned when the coordination context is created. That is, the

keys used to secure the channel may be independent of the key used to prove the right to register with the activity.

The security context MAY be re-established using the mechanisms described in WSTrust [WSTrust] and WS-SecureConversation [WSSecConv]. Similarly, secrets can be exchanged using the mechanisms described in WS-Trust. Note, however, that the current shared secret SHOULD NOT be used to encrypt the new shared secret.

Derived keys, the preferred solution from this list, can be specified using the mechanisms described in WS-SecureConversation. The following list summarizes common classes of attacks that apply to this protocol and identifies the mechanism to prevent/mitigate the attacks:

• **Message alteration** – Alteration is prevented by including signatures of the message information using WS-Security [WSSec].

• **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WS-Security.

• **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by comparing secured policies – see WS-Policy [WSPOLICY] and WSSecurityPolicy [WSSecPolicy]).

• **Authentication** – Authentication is established using the mechanisms described in WS-Security and WS-Trust [WSTrust]. Each message is authenticated using the mechanisms described in WS-Security [WSSec].

• **Accountability** – Accountability is a function of the type of and string of the key and algorithms being used. In many cases, a strong symmetric key provides sufficient accountability. However, in some environments, strong PKI

signatures are required.

• **Availability** – Many services are subject to a variety of availability attacks. Replay is a common attack and it is RECOMMENDED that this be addressed as described in the next bullet. Other attacks, such as network-level denial of service attacks are harder to avoid and are outside the scope of this specification.

That said, care should be taken to ensure that minimal processing be performed prior to any authenticating sequences.

• **Replay** – Messages may be replayed for a variety of reasons. To detect and eliminate this attack, mechanisms should be used to identify replayed messages such as the timestamp/nonce outlined in WS-Security [WSSec]. Alternatively, and optionally, other technologies, such as sequencing, can also be used to prevent replay of application messages.

8. Use of WS-Addressing Headers

The messages defined in WS-Coordination and WS-AtomicTransaction can be classified into three types:

• Request messages: **CreateCoordinationContext** and **Register**.

• Reply messages: **CreateCoordinationContextResponse** and **RegisterResponse**.

• Notification messages: **Commit**, **Rollback**, **Committed**, **Aborted**, **Prepare**, **Prepared**, **ReadOnly** and **Replay**.

Request and reply messages follow the standard "Request Reply" pattern as

defined in WS-Addressing. Notification messages follow the standard "one way" pattern as defined in WS-Addressing. There are two types of notification messages:

• A notification message is a terminal message when it indicates the end of a coordinator/participant relationship. **Committed**, **Aborted** and **ReadOnly** are terminal messages.

• A notification message is not a terminal message when it does not indicate the end of a coordinator/participant relationship. **Commit**, **Rollback**, **Prepare**, **Prepared** and **Replay** are not terminal messages. The following statements define addressing interoperability requirements for the respective WS-Coordination and WS-AtomicTransaction message types:

Request messages

• MUST include a wsa:MessageID header.

• MUST include a wsa:ReplyTo header.

Reply messages

• MUST include a wsa:RelatesTo header, specifying the MessageID from the corresponding Request message.

Non-terminal notification messages

• MUST include a wsa:ReplyTo header Terminal notification messages

• SHOULD NOT include a wsa:ReplyTo header Notification messages are addressed by both coordinators and participants using the Endpoint References initially obtained during the Register-RegisterResponse exchange. If a wsa:ReplyTo header is present in a notification message it MAY be used by the recipient, for example in cases where a Coordinator or Participant has forgotten

a transaction that is completed and needs to respond to a resent protocol message. Permanent loss of connectivity between a coordinator and a participant in an in-doubt state can result in data corruption.

All messages are delivered using connections initiated by the sender. Endpoint References MUST contain physical addresses and MUST NOT use well-known "anonymous" endpoint defined in WS-Addressing.

9. References

[KEYWORDS]

S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997

[SOAP]

W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000

[URI]

T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998

[XML-ns]

W3C Recommendation, "Namespaces in XML," 14 January 1999

[XML-Schema1]

W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001

[XML-Schema2]

W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001

[WSCOOR]

Web Services Coordination (WS-Coordination), Microsoft, IBM, and BEA Systems October 2004

[WSADDR]

Web Services Addressing (WS-Addressing), Microsoft, IBM, Sun, BEA Systems, SAP, Sun, August 2004

[WSPOLICY]

Web Services Policy Framework (WS-Policy), VeriSign, Microsoft, Sonic Software, IBM, BEA Systems, SAP, September 2004

[WSPOLICYASSERT]

Web Services Policy Assertions Language (WS-PolicyAssertions), Microsoft, IBM, BEA Systems, SAP, May 2003

[WSPOLICYATTACH]

Web Services Policy Attachment (WS-PolicyAttachment), VeriSign, Microsoft, Sonic Software, IBM, BEA Systems, SAP, September 2004

[WSDL]

Web Services Description Language (WSDL) 1.1 "http://www.w3.org/TR/2001/NOTE-wsdl-20010315"

[WSSec]

OASIS Standard 200401, March 2004, "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)"

[WSSecPolicy]

Web Services Security Policy Language (WS-SecurityPolicy), Microsoft, VeriSign, IBM, RSA Security, 18 December 2002

[WSSecConv]

Web Services Secure Conversation Language (WS-SecureConversation), OpenNetwork, Layer7, Netegrity, Microsoft, Reactivity, IBM, VeriSign, BEA Systems, Oblix, RSA Security, Ping Identity, Westbridge, Computer Associates, May 2004

[WSTrust]

Web Services Trust Language (WS-Trust), OpenNetwork, Layer7, Netegrity, Microsoft, Reactivity, VeriSign, IBM, BEA Systems, Oblix, RSA Security, Ping Identity, Westbridge, Computer Associates, May 2004

10. State Tables

The following state tables specify the behavior of coordinators and participants when presented with protocol messages or internal events. These tables present the view of a coordinator or participant with respect to a single partner. A coordinator with multiple participants can be understood as a collection of independent coordinator state machines. Each cell in the tables uses the following convention:

action to take

next state

Legend

Each state supports a number of possible events. Expected events are processed by taking the prescribed action and transitioning to the next state. Unexpected protocol messages will result in a fault message, with a standard fault code such as Invalid State or Inconsistent Internal State. Events that may not occur in a given state are labelled as N/A

Notes:

1. Transitions with a “N/A” as their action are inexpressible. A TM should view these
transitions as serious internal consistency issues, and probably fatal.

2. Internal events are those that are created either within a TM itself, or on its local
system.

3. “Forget” implies that the subordinate’s is participation is removed from the coordinator
(if necessary), and otherwise the message is ignored

Atomic Transaction 2PC protocol (Coordinator View)							
Inbound Events	**States**						
	None	**Active**	**Preparing**	**Prepared**	**PreparedSuccess**	**Committing**	**Aborting**
Register	*Invalid State* None	*Send RegisterResponse* Active	*Durable: Invalid State* Aborting Volatile: *Send RegisterResponse* Active	*N/A*	*Invalid State* PreparedSuccess	*Invalid State* Committing	*Invalid State* Aborting
Prepared	*Durable: Send Rollback Volatile: Invalid State* None	*Invalid State* Aborting	*Record Vote* Preparing	*N/A*	*Ignore* PreparedSuccess	*Resend Commit* Committing	*Resend Rollback, and forget* Aborting
ReadOnly	*Ignore* None	*Forget* Active	*Forget* Preparing	*N/A*	*Invalid State* PreparedSuccess	*Invalid State* Committing	*Forget* Aborting
Aborted	*Ignore* None	*Forget* Aborting	*Forget* Aborting	*N/A*	*Invalid State* PreparedSuccess	*Invalid State* Committing	*Forget* Aborting
Committed	*Ignore* None	*Invalid State* Aborting	*Invalid State* Aborting	*N/A*	*Invalid State* PreparedSuccess	*Forget* Committing	*Invalid State* Aborting
Replay	*Durable: Send Rollback Volatile: Invalid State* None	*Send Rollback* Aborting	*Send Rollback* Aborting	*N/A*	*Ignore* PreparedSuccess	*Send Commit* Committing	*Send Rollback* Aborting
Internal Events							
User Commit	*Return Aborted* None	*Send Prepare* Preparing	*Ignore* Preparing	*N/A*	*Ignore* PreparedSuccess	*Return Committed* Committing	*Return Aborted* Aborting
User Rollback	*Return Aborted* None	*Send Rollback* Aborting	*Send Rollback* Aborting	*N/A*	*Invalid State* PreparedSuccess	*Invalid State* Committing	*Return Aborted* Aborting
Expires Times out	*N/A*	*Send Rollback* Aborting	*Send Rollback* Aborting	*N/A*	*Ignore* PreparedSuccess	*Ignore* Committing	*Ignore* Aborting
Comms Times out	*N/A*	*N/A*	*Resend Prepare* Preparing	*N/A*	*N/A*	*Resend Commit* Committing	*N/A*
Commit Decision	*N/A*	*N/A*	*Record Outcome* PreparedSuccess	*N/A*	*N/A*	*N/A*	*N/A*
Write Done	*N/A*	*N/A*	*N/A*	*N/A*	*Send Commit* Committing	*N/A*	*N/A*
Write Failed	*N/A*	*N/A*	*N/A*	*N/A*	*Send Rollback* Aborting	*N/A*	*N/A*
All Forgotten	*N/A*	Active	None	*N/A*	*N/A*	None	None

Atomic Transaction 2PC protocol (Participant View)							
Inbound Events	**States**						
	None	**Active**	**Preparing**	**Prepared**	**PreparedSuccess**	**Committing**	**Aborting**
Register Response	*Register Subordinate* Active	*Invalid State* Active	*Invalid State* Aborting	*Invalid State* Prepared	*Invalid State* PreparedSuccess	*Invalid State* Committing	*Invalid State* Aborting
Prepare	*Send Aborted* None	*Gather Vote Decision* Preparing	*Ignore* Preparing	*Ignore* Prepared	*Resend Prepared* PreparedSuccess	*Ignore* Committing	*Resend Aborted, and forget* Aborting
Commit	*Send Committed* None	*Invalid State* Aborting	*Invalid State* Aborting	*Invalid State* Aborting	*Initiate commit decision* Committing	*Ignore* Committing	*InconsistentInternalStat* Aborting
Rollback	*Send Aborted* None	*Initiate Rollback, Send Aborted, and Forget* Aborting	*Initiate Rollback, Send Aborted, and Forget* Aborting	*Initiate Rollback, Send Aborted, and Forget* Aborting	*Initiate Rollback, Send Aborted, and Forget* Aborting	*InconsistentInternalState* Committing	*Send Aborted, and Forget* Aborting
Internal Events							
Expires Times out	*N/A*	*Send Aborted* Aborting	*Send Aborted* Aborting	*Ignore* Prepared	*Ignore* PrepareSuccess	*Ignore* Committing	*Ignore* Aborting
Comms Times out	*N/A*	*N/A*	*N/A*	*N/A*	*Resend Prepared* PreparedSuccess	*N/A*	*N/A*
Commit Decision	*N/A*	*N/A*	*Record Commit* Prepared	*N/A*	*N/A*	*Send Committed and Forget* Committing	*N/A*
Rollback Decision	*N/A*	*N/A*	*Send Aborted* Aborting	*N/A*	*N/A*	*N/A*	*N/A*
Write Done	*N/A*	*N/A*	*N/A*	*Send Prepared* PreparedSuccess	*N/A*	*N/A*	*N/A*
Write Failed	*N/A*	*N/A*	*N/A*	*Initiate Rollback, Send Aborted, and Forget* Aborting	*N/A*	*N/A*	*N/A*
All Forgotten	None	*N/A*	*Send ReadOnly* None	*N/A*	*N/A*	None	None

Decision

Notes:

1. Transitions with a “N/A” as their action are inexpressible. A TM should view these transitions as serious internal consistency issues, and probably fatal.

2. Internal events are those that are created either within a TM itself, or on its local system.

ANEXO D - WEB SERVICES COORDINATION (WS-COORDINATION)

November 2004

Authors

Luis Felipe Cabrera, Microsoft

George Copeland, Microsoft

Max Feingold, Microsoft

Tom Freund, IBM

Jim Johnson, Microsoft

Chris Kaler, Microsoft

Johannes Klein, Microsoft

David Langworthy, Microsoft (Editor)

Anthony Nadalin, IBM

David Orchard, BEA Systems

Ian Robinson, IBM

John Shewchuk, Microsoft

Tony Storey, IBM

Copyright Notice

Services Coordination" Specification that you make:

1. A link or URL to the "Web Services Coordination" Specification at one of the Authors' websites

2. The copyright notice as shown in the "Web Services Coordination " Specification.

IBM, Microsoft and BEA (collectively, the "Authors") each agree to grant you a license, under royalty-free and otherwise reasonable, non-discriminatory terms and conditions, to their respective essential patent claims that they deem necessary to implement the "Web Services Coordination" Specification.

THE "WEB SERVICES COORDINATION" SPECIFICATION IS PROVIDED "AS IS," AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE "WEB SERVICES COORDINATION" SPECIFICATION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE "WEB SERVICES COORDINATION" SPECIFICATION.

The name and trademarks of the Authors may NOT be used in any manner, including advertising or publicity pertaining to the "Web Services Coordination"

Specification or its contents without specific, written prior permission. Title to copyright in the “Web Services Coordination” Specification will at all times remain with the Authors. No other rights are granted by implication, estoppel or otherwise.

Abstract

This specification (WS-Coordination) describes an extensible framework for providing protocols that coordinate the actions of distributed applications. Such coordination protocols are used to support a number of applications, including those that need to reach consistent agreement on the outcome of distributed activities. The framework defined in this specification enables an application service to create a context needed to propagate an activity to other services and to register for coordination protocols. The framework enables existing transaction processing, workflow, and other systems for coordination to hide their proprietary protocols and to operate in a heterogeneous environment.

Additionally this specification describes a definition of the structure of context and the requirements for propagating context between cooperating services.

Composable Architecture

By using the SOAP [SOAP] and WSDL [WSDL] extensibility model, SOAP-based and WSDL-based specifications are designed to be composed with each other to define a rich Web services environment. As such, WS-Coordination by itself does not define all the features required for a complete solution. WS-Coordination is a building block that is used in conjunction with other specifications and application-specific protocols to accommodate a wide variety of protocols related to the operation of distributed Web services.

The Web service protocols defined in this specification should be used when interoperability is needed across vendor implementations, trust domains, etc. Thus, the Web service protocols defined in this specification can be combined with proprietary protocols within the same application.

Status

WS-Coordination and related specifications are provided for use as-is and for review and evaluation only. Microsoft, BEA, and IBM will solicit your contributions and suggestions in the near future. Microsoft, BEA, and IBM make no warrantees or representations regarding the specification in any manner whatsoever.

Acknowledgments

The following individuals have provided invaluable input into the design of the WSCoordination specification:

Francisco Curbera, IBM

Sanjay Dalal, BEA Systems

Doug Davis, IBM

Don Ferguson, IBM

Kirill Garvylyuk, Microsoft

Frank Leymann, IBM

Thomas Mikalsen, IBM

Jagan Peri, Microsoft

Alex Somogyi, BEA Systems

Stefan Tai, IBM

Satish Thatte, Microsoft

Sanjiva Weerawarana, IBM

We also wish to thank the technical writers and development reviewers who provided feedback to improve the readability of the specification.

Table of Contents

1. Introduction

The current set of Web service specifications [WSDL, SOAP] defines protocols for Web service interoperability. Web services increasingly tie together a large number of participants forming large distributed computational units – we refer to these computation units as activities.

The resulting activities are often complex in structure, with complex relationships between their participants. The execution of such activities often takes a long time to complete due to business latencies and user interactions.

This specification defines an extensible framework for coordinating activities using a coordinator and set of coordination protocols. This framework enables participants to reach consistent agreement on the outcome of distributed activities. The coordination protocols that can be defined in this framework can

accommodate a wide variety of activities, including protocols for simple short-lived operations and protocols for complex long-lived business activities.

Note that the use of the coordination framework is not restricted to transaction processing systems; a wide variety of protocols can be defined for distributed applications.

1.1 The Model

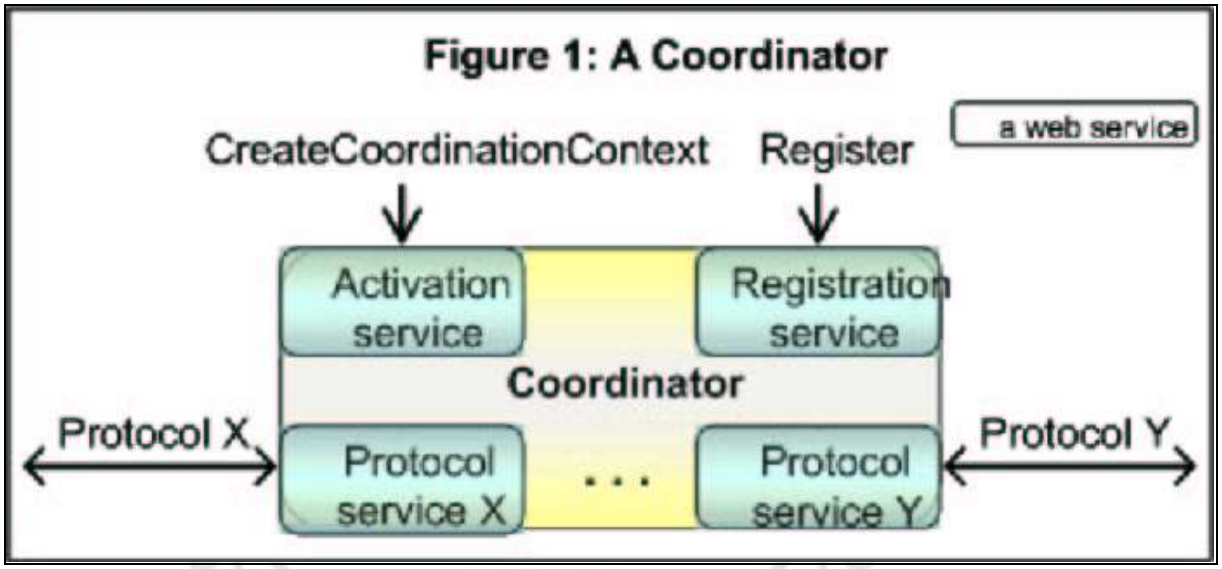
This specification describes a framework for a coordination service (or coordinator) which consists of these component services:

• An Activation service with an operation that enables an application to create a coordination instance or context.

• A Registration service with an operation that enables an application to register for coordination protocols.

• A coordination type-specific set of coordination protocols.

This is illustrated below in Figure 1.



Applications use the Activation service to create the coordination context for an activity.

Once a coordination context is acquired by an application, it is then sent by whatever appropriate means to another application.

The context contains the necessary information to register into the activity specifying the coordination behavior that the application will follow.

Additionally, an application that receives a coordination context may use the Registration service of the original application or may use one that is specified by an interposing, trusted coordinator. In this manner an arbitrary collection of Web services may coordinate their joint operation.

1.2 Extensibility

The specification provides for extensibility and flexibility along two dimensions.

The framework allows for:

• The publication of new coordination protocols.

• The selection of a protocol from a coordination type and the definition of extension elements that can be added to protocols and message flows.

Extension elements can be used to exchange application-specific data on top of message flows already defined in this specification. This addresses the need to exchange such data as isolation-level supported signatures or other information related to businesslevel coordination protocols. The data can be logged for auditing purposes, or evaluated to ensure that a decision meets certain business-specific constraints. To understand the syntax used in this specification, you should be familiar with the WSDL [WSDL] specification, including its HTTP and SOAP binding styles. All WSDL port type definitions provided here assume the existence of corresponding SOAP and HTTP bindings.

Terms introduced in this specification are explained in the body of the specification and summarized in the glossary.

1.3 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [KEYWORDS].

Namespace URIs of the general form "some-URI" represents some applicationdependent or context-dependent URI as defined in RFC2396 [URI].

This specification uses an informal syntax to describe the XML grammar of the XML fragments below:

• The syntax appears as an XML instance, but the values indicate the data types instead of values.

• Element names ending in "..." (such as <element.../> or <element...>) indicate that elements/attributes irrelevant to the context are being omitted.

• Attributed names ending in "..." (such as name=...) indicate that the values are specified below.

• Grammar in bold has not been introduced earlier in the document, or is of particular interest in an example.

• <-- description --> is a placeholder for elements from some "other" namespace (like ##other in XSD).

• Characters are appended to elements, attributes, and <!-- descriptions --> as follows: "?" (0 or 1), "*" (0 or more), "+" (1 or more). The characters "[" and "]" are used to indicate that contained items are to be treated as a group with respect to the "?", "*", or "+" characters.

• The XML namespace prefixes (defined below) are used to indicate the namespace of the element being defined.

• Examples starting with <?xml contain enough information to conform to this specification; others examples are fragments and require additional information to be specified in order to conform. XSD schemas and WSDL definitions are provided as a formal definition of grammars [xml-schema1] [WSDL].

1.4 Namespace

The XML namespace [XML-ns] URI that MUST be used by implementations of

this specification is:

http://schemas.xmlsoap.org/ws/2004/10/wscoor

The namespace prefix "wscoor" used in this specification is associated with this URI. The following namespaces are used in this document:

Prefix Namespace

S http://www.w3.org/2001/12/soap-envelope

wscoor http://schemas.xmlsoap.org/ws/2004/10/wscoor

wsa http://schemas.xmlsoap.org/ws/2004/08/addressing

If an action URI is used, then the action URI MUST consist of the coordination namespace URI concatenated with the '/' character and the element name. For example:

http://schemas.xmlsoap.org/ws/2004/10/wscoor/Register

1.5 XSD and WSDL Files

The following links hold the XML schema and the WSDL declarations defined in this

document.

http://schemas.xmlsoap.org/ws/2004/10/wscoor/wscoor.xsd

http://schemas.xmlsoap.org/ws/2004/10/wscoor/wscoor.wsdl

Soap bindings for the WSDL documents defined in this specification MUST use "document" for the *style* attribute.

2. CoordinationContext

The CoordinationContext is a Context type, as described in the Appendix A, that is used to pass Coordination information to parties involved in a coordination service. CoordinationContext elements are placed within application messages. Conveying a context on an application message is commonly referred to as flowing the context. A CoordinationContext provides access to a coordination registration service, a coordination type, and relevant extensions. The following is an example of a CoordinationContext supporting a transaction service:

<?xml version="1.0" encoding="utf-8"?>

<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"

<S:Header>

. . .

<wscoor:CoordinationContext

xmlns:wscoor="http://schemas.xmlsoap.org/ws/2004/10/wscoor"

xmlns:myApp="http://fabrikam123.com/myApp"

S:mustUnderstand="true">

<wscoor:Expires>3000</wscoor:Expires>

<wscoor:Identifier>

http://Fabrikam123.com/SS/1234

</wscoor:Identifier>

<wscoor:CoordinationType>

http://schemas.xmlsoap.org/ws/2004/10/wsat

</wscoor:CoordinationType>

<wscoor:RegistrationService>

<wsa:Address>

http://Business456.com/mycoordinationservice/registration

</wsa:Address>

<wsa:ReferenceProperties>

<myApp:BetaMark> ... </myApp:BetaMark>

<myApp:EBDCode> ... </myApp:EBDCode>

</wsa:ReferenceProperties>

</wscoor:RegistrationService>

<myApp:IsolationLevel>

RepeatableRead

</myApp:IsolationLevel>

</wscoor:CoordinationContext>

. . .

</S:Header>

When an application propagates an activity using a coordination service, applications MUST include a Coordination context in the outgoing message. When a context is exchanged as a SOAP header, the mustUnderstand attribute must be present and its value must be true.

3. Coordination Service

The Coordination service (or coordinator) is an aggregation of the following services:

• Activation service: Defines a CreateCoordinationContext operation that allows a CoordinationContext to be created. The exact semantics are defined in the specification that defines the coordination type. The Coordination service MAY support the Activation service.

• Registration service: Defines a Register operation that allows a Web service to register to participate in a coordination protocol. The Coordination service MUST support the Registration service.

• A set of coordination protocol services for each supported coordination type. These are defined in the specification that defines the coordination type.

Figure 2 illustrates how two application services (App1 and App2) with their own coordinators (CoordinatorA and CoordinatorB) interact as the activity propagates between them. The protocol Y and services Ya and Yb are specific to a coordination type, which are not defined in this specification.

1. App1 sends a CreateCoordinationContext for coordination type Q, getting back a Context Ca that contains the activity identifier A1, the coordination type Q and an Endpoint Reference to CoordinatorA's Registration service RSa.

2. App1 then sends an application message to App2 containing the Context Ca.

3. App2 prefers CoordinatorB, so it uses CreateCoordinationContext with Ca as an input to interpose CoordinatorB. CoordinatorB creates its own CoordinationContext Cb that contains the same activity identifier and coordination type as Ca but with its

own Registration service RSb.

4. App2 determines the coordination protocols supported by the coordination

type Q and then Registers for a coordination protocol Y at CoordinatorB, exchanging Endpoint References for App2 and the protocol service Yb. This forms a logical connection between these Endpoint References that the protocol Y can use.

5. This registration causes CoordinatorB to forward the registration onto CoordinatorA's Registration service RSa, exchanging Endpoint References for Yb and the protocol service Ya. This forms a logical connection between these Endpoint References that the protocol Y can use.

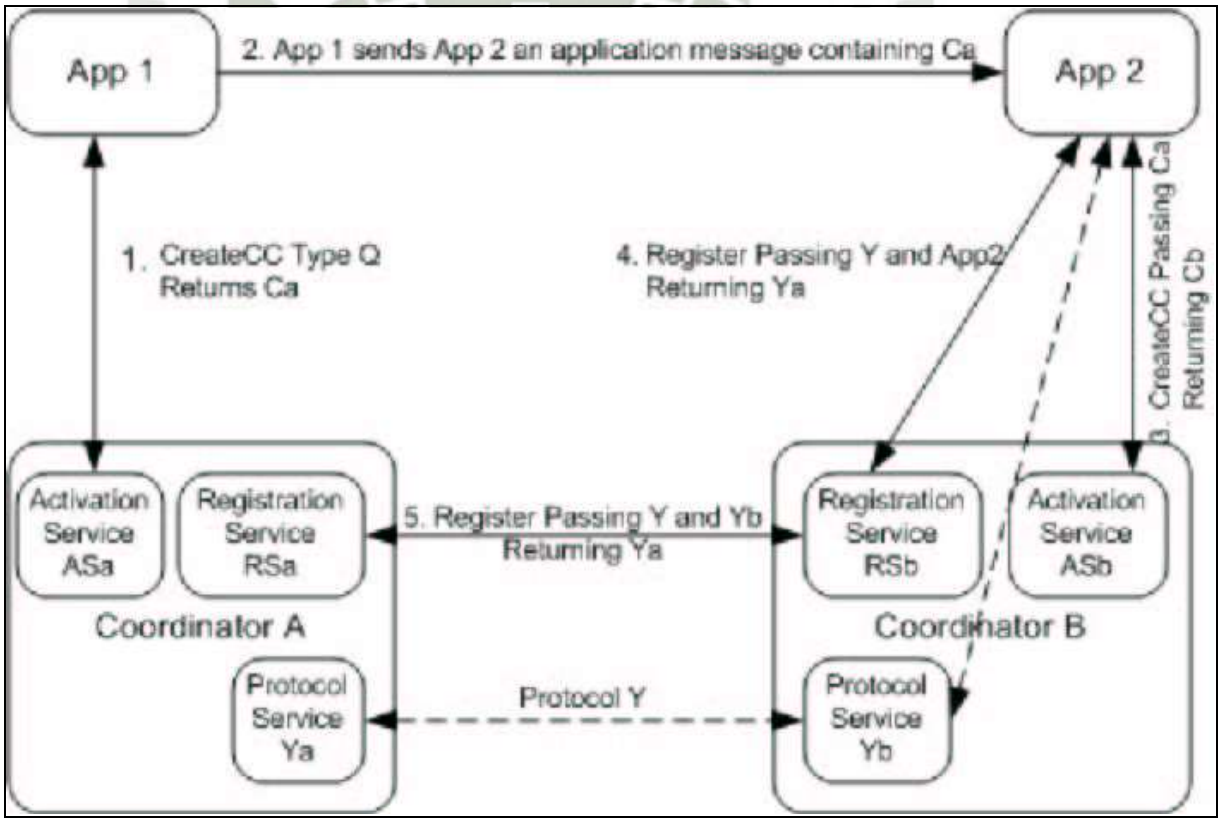


Figure 2: Two applications with their own coordinators

3.1 Activation Service

The Activation service creates a new activity and returns its coordination context. An application sends:

CreateCoordinationContext

The structure and semantics of this message is defined in Section 3.1.1. The activation service returns:

CreateCoordinationContextResponse

The structure and semantics of this message is defined in Section 3.1.2.

3.1.1 CreateCoordinationContext

This request is used to create a coordination context that supports a coordination type (i.e., a service that provides a set of coordination protocols). This command is required when using a network-accessible Activation service in heterogeneous environments that span vendor implementations. To fully understand the semantics of this operation it is necessary to read the specification where the coordination type is defined (e.g. WSAtomicTransaction). The following pseudo schema defines this element:

<CreateCoordinationContext ...>

<CoordinationType> ... </CoordinationType>

<Expires> ... </Expires>?

<CurrentContext> ... </CurrentContext>?

...

</CreateCoordinationContext>

/CreateCoordinationContext/CoordinationType

This provides the unique identifier for the desired coordination type for the activity

(e.g., a URI to the Atomic Transaction coordination type).

/CreateCoordinationContext/Expires

Optional. The expiration for the returned CoordinationContext expressed as an unsigned integer in milliseconds.

/CreateCoordinationContext/CurrentContext

Optional. The current CoordinationContext. This may be used for a variety of purposes including recovery and subordinate coordination environments.

/CreateCoordinationContext /{any}

Extensibility elements may be used to convey additional information.

/CreateCoordinationContext /@{any}

Extensibility attributes may be used to convey additional information.

A CreateCoordinationContext message can be as simple as the following example.

<CreateCoordinationContext>

<CoordinationType>

http://schemas.xmlsoap.org/ws/2004/10/wsat

</CoordinationType>

</CreateCoordinationContext>

3.1.2 CreateCoordinationContextResponse

This returns the CoordinationContext that was created.

The following pseudo schema defines this element:

<CreateCoordinationContextResponse ...>

<CoordinationContext> ... </CoordinationContext>

...

</CreateCoordinationContextResponse>

/CreateCoordinationContext/CoordinationContext

This is the created coordination context.

/CreateCoordinationContext /{any}

Extensibility elements may be used to convey additional information.

/CreateCoordinationContext /@{any}

Extensibility attributes may be used to convey additional information.

The following example illustrates a response:

<CreateCoordinationContextResponse>

<CoordinationContext>

<Identifier>

http://Business456.com/tm/context1234

</Identifier>

<CoordinationType>

http://schemas.xmlsoap.org/ws/2004/10/wsat

</CoordinationType>

<RegistrationService>

<wsa:Address>

http://Business456.com/tm/registration

</wsa:Address>

<wsa:ReferenceProperties>

<myapp:PrivateInstance>

1234

</myapp:PrivateInstance>

</wsa:ReferenceProperties>

</RegistrationService>

</CoordinationContext>

</CreateCoordinationContextResponse>

3.2 Registration Service

Once an application has a coordination context from its chosen coordinator, it can register for the activity. The interface provided to an application registering for an activity and for an interposed coordinator registering for an activity is the same. The requester sends:

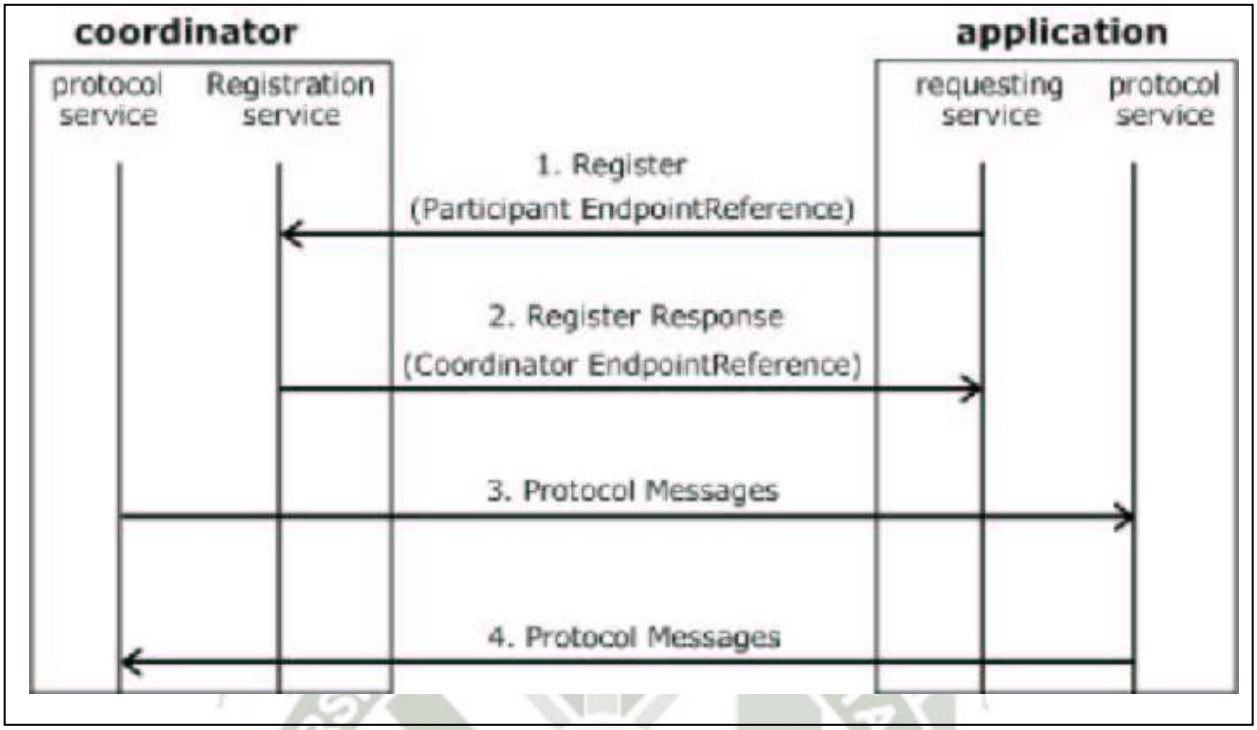
Register

The syntax and semantics of this message are defined in Section 3.2.1.

The coordinator's registration service responds with: Registration Response

The syntax and semantics of this message are defined in Section 3.2.2.

Figure 3: The usage of Endpoint References during registration



In Figure 3, the coordinator provides the Registration Endpoint Reference in the CoordinationContext during the CreateCoordinationContext operation. The requesting service receives the Registration service Endpoint Reference in the CoordinationContext in an application message.

1.) The Register message targets this Endpoint Reference and includes the participant protocol service Endpoint Reference as a parameter.

2.) The RegisterResponse includes the coordinator's protocol service Endpoint Reference.

3. & 4.) At this point, both sides have the Endpoint References of the other's protocol service, so the protocol messages can target the other side. These Endpoint References may contain (opaque) wsa:ReferenceProperties to fully qualify the target protocol service endpoint. According to the mapping rules

defined in the WS-Addressing specification, all such reference properties must be copied literally as headers in any message targeting the endpoint.

3.2.1 Register Message

The Register request is used to do the following:

• Participant selection and registration in a particular Coordination protocol under the current coordination type supported by the Coordination Service.

• Exchange Endpoint References. Each side of the coordination protocol (participant and coordinator) supplies a Endpoint Reference. Participants can register for multiple Coordination protocols by issuing multiple Register operations. WS-Coordination assumes that transport protocols provide for message batching if required.

The following pseudo schema defines this element:

<Register ...>

<ProtocolIdentifier> ... </ProtocolIdentifier>

<ParticipantProtocolService> ... </ParticipantProtocolService>

...

</Register>

/Register/ProtocolIdentifier

This URI provides the identifier of the coordination protocol selected for registration.

/Register/ParticipantProtocolService

The Endpoint Reference that the registering participant wants the coordinator to use

for the Coordination protocol (See WS-Addressing [WSADDR]).

/Register/{any}

Extensibility elements may be used to convey additional information.

/ Register/@{any}

Extensibility attributes may be used to convey additional information.

The following is an example registration message:

<Register>

<ProtocolIdentifier>

http://schemas.xmlsoap.org/ws/2004/10/wsat/Volatile2PC

</ProtocolIdentifier>

<ParticipantProtocolService>

<wsa:Address>

"http://Adventure456.com/participant2PCservice"

</wsa:Address>

<wsa:ReferenceProperties>

<BetaMark> AlphaBetaGamma </BetaMark>

</wsa:ReferenceProperties>

</ParticipantProtocolService>

</Register>

3.2.2 RegistrationResponse Message

The response to the registration message contains the coordinators Endpoint

Reference.

The following pseudo schema defines this element:

<RegisterResponse ...>

<CoordinatorProtocolService> ... </CoordinatorProtocolService>

...

</RegisterResponse>

/RegisterResponse/CoordinatorProtocolService

The Endpoint Reference that the Coordination service wants the registered participant to use for the Coordination protocol.

/RegisterResponse/{any}

Extensibility elements may be used to convey additional information.

/RegisterResponse /@{any}

Extensibility attributes may be used to convey additional information. The following is an example of a RegisterResponse message:

<RegisterResponse>

<CoordinatorProtocolService>

<wsa:Address>

"http://Business456.com/mycoordinationservice/coordinator" />

<wsa:ReferenceProperties>

<myapp:MarkKey> %%F03CA2B%% </myapp:MarkKey>

</wsa:ReferenceProperties>

</CoordinatorProtocolService>

</RegisterResponse>

4. Coordination Faults

This section defines well-known error codes to be used in conjunction with an underlying fault handling mechanism. For example, when used with SOAP 1.2 the identifier code is the fault sub-code and any additional information is passed in the detail.

4.1. Invalid State

This fault is sent by either the coordinator or a participant to indicate that the endpoint that generates the fault has entered an invalid state. This is an unrecoverable condition. The qualified name of the fault code is:

wscoor:InvalidState

4.2. Invalid Protocol

This fault is sent by either the coordinator or a participant to indicate that the endpoint that generates the fault received a message from an invalid protocol. This is an unrecoverable condition.

The qualified name of the fault code is:

wscoor:InvalidProtocol

4.3. Invalid Parameters

This fault is sent by either the coordinator or a participant to indicate that the endpoint that generated the fault received invalid parameters on a within a message. This is an unrecoverable condition. The qualified name of the fault

code is:

wscoor:InvalidParameters

4.4. No Activity

This fault is sent by the coordinator if the participant has been quiet for too long and is presumed to have ended.

The qualified name of the fault code is:

wscoor:NoActivity

4.5. Context Refused

This fault is sent to a coordinator to indicate that the endpoint cannot accept a context which it was passed:

The qualified name of the fault code is:

wscoor:ContextRefused

4.6 Already Registered

This fault is sent to a participant if the coordinator detects that the participant attempted to register for the same protocol of the same activity more than once.

wscoor:AlreadyRegistered

5. Security Model

The primary goals of security with respect to WS-Coordination are to:

1. ensure only authorized principals can create coordination contexts

2. ensure only authorized principals can register with an activity

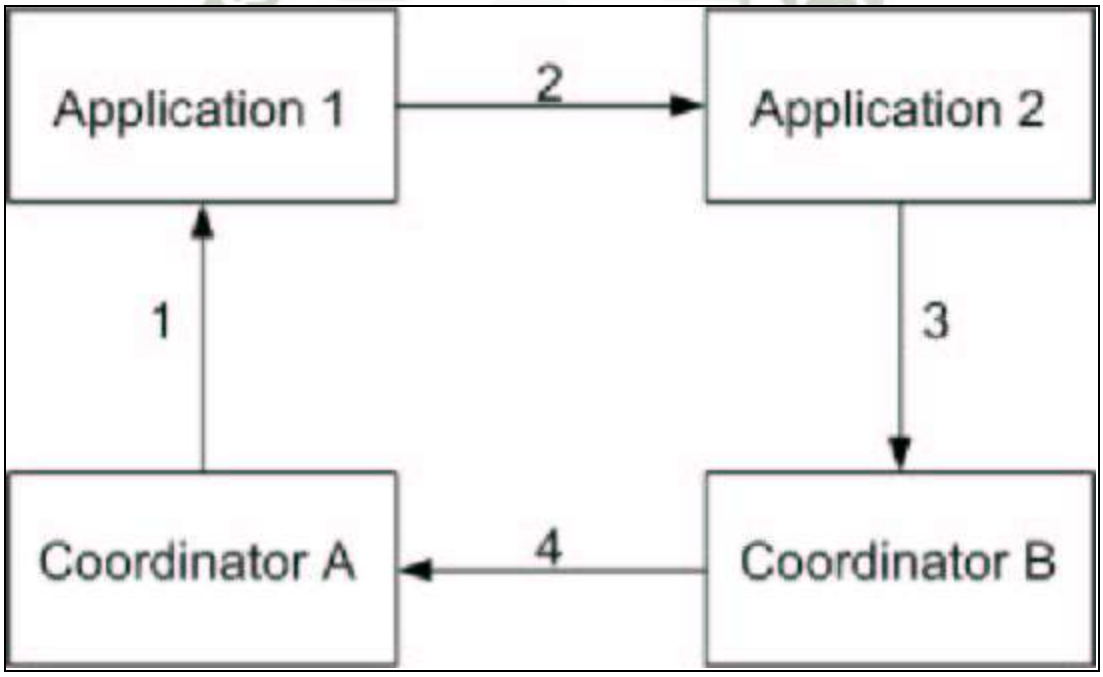
3. ensure only legitimate coordination contexts are used to register

4. enable existing security infrastructures to be leveraged

5. allow principal authorization to be based on federated identities

These goals build on the general security requirements for integrity, confidentiality, and authentication, each of which is provided by the foundations built using the Web service security specifications such as WS-Security [WSSec] and WS-Trust [WSTrust].

The following figure illustrates a fairly common usage scenario:



In the figure above, step 1 involves the creation and subsequent communication between the creator of the context and the coordinator A (root). It should be noted that this may be a private or local communication. Step 2 involves the

delegation of the right to register with the activity using the information from the coordination context and subsequent application messages between two applications (and may include middleware involvement) which are participants in the activity. Step 3 involves delegation of the right to register with the activity to coordinator B (subordinate) that manages all access to the activity on behalf of the second, and possibly other parties.

Again note that this may also be a private or local communication. Step 4 involves

registration with the coordinator A by the coordinator B and proof that registration rights were delegated.

It should be noted that many different coordination topologies may exist which may leverage different security technologies, infrastructures, and token formats. Consequently an appropriate security model must allow for different topologies, usage scenarios, delegation requirements, and security configurations. To achieve these goals, the security model for WS-Coordination leverages the infrastructure provided by WS-Security [WSSec], WS-Trust [WSTrust], WS-Policy [WSPOLICY], and WS-SecureConversation [WSSecConv]: Services have policies specifying their requirements and requestors provide claims (either implicit or explicit) and the requisite proof of those claims. There are a number of different mechanisms which can be used to affect the previously identified goals. However, this specification RECOMMENDS a simple mechanism, which is described here, for use in interoperability scenarios.

5.1. CoordinationContext Creation

When a coordination context is created (step 1 above) the message is secured using the mechanisms described in WS-Security. If the required claims are proven, as described by WS-Policy [WSPOLICY], then the coordination context is created. A set of claims, bound to the identity of the coordination context's creator, and maintained by the coordinator, are associated with the creation of the coordination context. The creator of the context must obtain these claims from the coordinator.

Before responding with the claims, the coordinator requires proof of the requestor's identity.

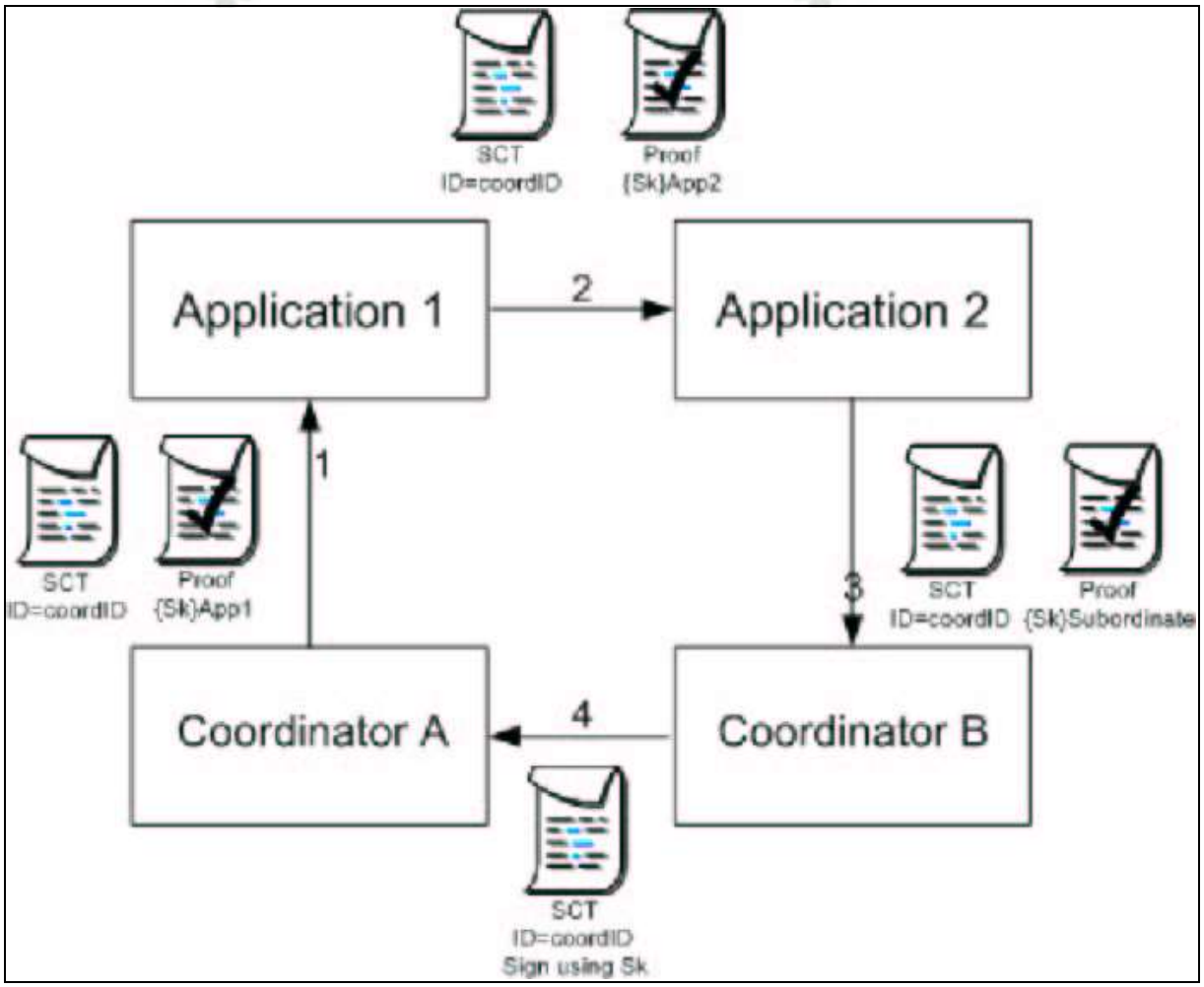
Additionally, the coordinator provides a shared secret which is used to indicate authorization to register with the coordination context by other parties. The secret is communicated using a security context token (SCT) and <wsse:RequestSecurityTokenResponse> element using the mechanisms described in WS-SecureConversation [WSSecConv] and WS-Trust [WSTrust]. This secret may be delegated to other parties as described in the next section.

5.2. Registration Rights Delegation

Secret delegation is performed by propagation of the SCT that was created by the root Coordinator. This involves using the <wsse:RequestSecurityTokenResponse> element and encrypting the shared

secret for the new participant as a new proof token. The participants can then use the SCT (which has a URI that can be used to locate the corresponding key/secret) using WS-Security by providing a signature based on the key/secret to authenticate and authorize the right to register with the activity that created the coordination context.

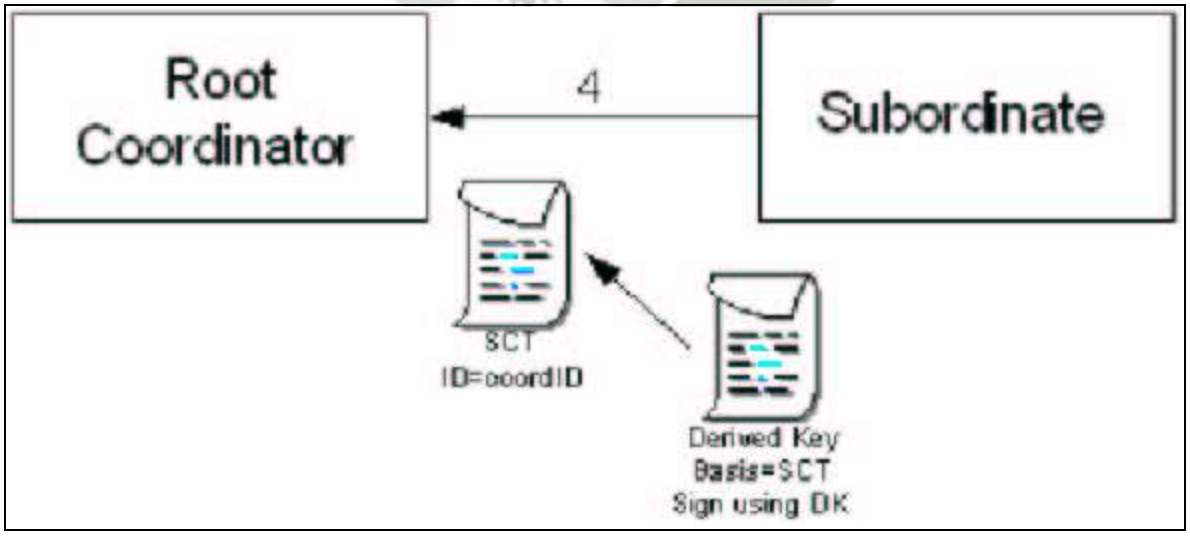
The figure below illustrates this simple key delegation model:



As illustrated in the figure above, the coordinator A, root in this case, (or its delegate) creates a security context token (cordID) representing the right to

register and returns (using the mechanisms defined in WS-Trust [WSTrust]) that token to Application 1 (or its delegate) (defined in WS-SecureConversation [WSSecConv]) and a session key (Sk) encrypted for Application 1 inside of a proof token. This key allows Application 1 (or its delegate) to prove it is authorized to use the SCT. Application 1 (or its delegate) decrypts the session key (Sk) and encrypts it for Application 2 its delgate. Application 2 (or its delegate) performs the same act encrypting the key for the subordinate. Finally, coordinator B, subordinate in this case, proves its right to the SCT by including a signature using Sk.

It is RECOMMENDED by this specification that the key/secret never actually be used to secure a message. Instead, keys derived from this secret SHOULD be used to secure a message, as described in WS-SecureConversation [WSSecConv]. This technique is used to maximize the strength of the key/secret as illustrated in the figure below:



6. Security Considerations

It is strongly RECOMMENDED that the communication between services be secured using the mechanisms described in WS-Security [WSSec]. In order to properly secure messages, the body and all relevant headers need to be included in the signature.

Specifically, the <wscoor:CoordinationContext> header needs to be signed with the body and other key message headers in order to "bind" the two together. This will ensure that the coordination context is not tampered. In addition the reference properties within an Endpoint Reference may be encrypted to ensure their privacy. In the event that a participant communicates frequently with a coordinator, it is RECOMMENDED that a security context be established using the mechanisms described in WS-Trust [WSTrust] and WS-SecureConversation [WSSecConv] allowing for potentially more efficient means of authentication.

It is common for communication with coordinators to exchange multiple messages. As a result, the usage profile is such that it is susceptible to key attacks. For this reason it is strongly RECOMMENDED that the keys used to secure the channel be changed frequently. This "re-keying" can be effected a number of ways. The following list outlines four common techniques:

• Attaching a nonce to each message and using it in a derived key function with the shared secret

• Using a derived key sequence and switch "generations"

• Closing and re-establishing a security context

• Exchanging new secrets between the parties

It should be noted that the mechanisms listed above are independent of the SCT and secret returned when the coordination context is created. That is, the keys used to secure the channel may be independent of the key used to prove the right to register with the coordination context.

The security context MAY be re-established using the mechanisms described in WS-Trust [WSTrust] and WS-SecureConversation [WSSecConv]. Similarly, secrets can be exchanged using the mechanisms described in WS-Trust. Note, however, that the current shared secret SHOULD NOT be used to encrypt the new shared secret. Derived keys, the preferred solution from this list, can be specified using the mechanisms described in WS-SecureConversation.

The following list summarizes common classes of attacks that apply to this protocol and identifies the mechanism to prevent/mitigate the attacks:

• **Message alteration** – Alteration is prevented by including signatures of the message information using WS-Security [WSSec].

• **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WS-Security.

• **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by comparing secured policies – see WS-Policy [WSPOLICY] and WSSecurityPolicy [WSSecPolicy]).

• **Authentication** – Authentication is established using the mechanisms described in WS-Security [WSSec] and WS-Trust [WSTrust]. Each message is authenticated using the mechanisms described in WS-Security.

• **Accountability** – Accountability is a function of the type of and string of the key and algorithms being used. In many cases, a strong symmetric key provides sufficient accountability. However, in some environments, strong PKI signatures are required.

• **Availability** – Many services are subject to a variety of availability attacks. Replay is a common attack and it is RECOMMENDED that this be addressed as described in the next bullet. Other attacks, such as network-level denial of service attacks are harder to avoid and are outside the scope of this specification. That said, care should be taken to ensure that minimal processing be performed prior to any authenticating sequences.

• **Replay** – Messages may be replayed for a variety of reasons. To detect and eliminate this attack, mechanisms should be used to identify replayed messages such as the timestamp/nonce outlined in WS-Security [WSSec]. Alternatively, and optionally, other technologies, such as sequencing, can also

be used to prevent replay of application messages.

7. Glossary

The following definitions are used throughout this specification:

Activation service: This supports a CreateCoordinationContext operation that is used by participants to create a CoordinationContext.

CoordinationContext: Contains the activity identifier, its coordination type that represents the collection of behaviors supported by the activity and a Registration service Endpoint Reference that participants can use to register for one or more of the protocols supported by that activity's coordination type.

Coordination protocol: The definition of the coordination behavior and the messages exchanged between the coordinator and a participant playing a specific role within a coordination type. WSDL definitions are provided, along with sequencing rules for the messages. The definition of coordination protocols are provided in additional specification (e.g., WS-AtomicTransaction).

Coordination type: A defined set of coordination behaviors, including how the service accepts context creations and coordination protocol registrations, and drives the coordination protocols associated with the activity.

Coordination service (or Coordinator): This service consists of an activation service, a registration service, and a set of coordination protocol services.

Participant: A service that is carrying out a computation within the activity. A participant receives the CoordinationContext and can use it to register for coordination protocols.

Registration service: This supports a Register operation that is used by participants to register for any of the coordination protocols supported by a coordination type, such as Atomic Transaction 2PC or Business Agreement NestedScope.

Web service: A Web service is a computational service, accessible via messages of definite, programming-language-neutral and platform-neutral format, and which has no special presumption that the results of the computation are used primarily for display by a user-agent.

8. References

[KEYWORDS]

S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997

[SOAP]

W3C Note, "Simple Object Access Protocol (SOAP) 1.1," 08 May 2000

[URI]

T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998

[XML-ns]

W3C Recommendation, "Namespaces in XML," 14 January 1999

[XML-Schema1]

W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001

[XML-Schema2]

W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001

[WSADDR]

Web Services Addressing (WS-Addressing), Microsoft, IBM, Sun, BEA Systems, SAP, Sun, August 2004

[WSPOLICY]

Web Services Policy Framework (WS-Policy), VeriSign, Microsoft, Sonic Software, IBM, BEA Systems, SAP, September 2004

[WSSec]

OASIS Standard 200401, March 2004, "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)"

[WSSecPolicy]

Web Services Security Policy Language (WS-SecurityPolicy), Microsoft, VeriSign, IBM, and RSA Security Inc., 18 December 2002

[WSSecConv]

Web Services Secure Conversation Language (WS-SecureConversation), OpenNetwork, Layer7, Netegrity, Microsoft, Reactivity, IBM, VeriSign, BEA Systems, Oblix, RSA Security, Ping Identity, Westbridge, Computer Associates, May 2004

[WSTrust]

Web Services Trust Language (WS-Trust), OpenNetwork, Layer7, Netegrity, Microsoft, Reactivity, VeriSign, IBM, BEA Systems, Oblix, RSA Security, Ping Identity, Westbridge, Computer Associates, May 2004

[WSAT]

Web Services AtomicTransactions (WS-AtomicTransactions), Microsoft, IBM, and BEA Systems, October 2004

[WSDL]

Web Services Description Language (WSDL) 1.1 "http://www.w3.org/TR/2001/NOTEwsdl- 20010315"

ANEXO E - SISTEMATIZACIÓN DE LOS RESULTADOS DE LA ENCUESTA

Cuadros de elaboración propia:

1. ¿Cómo considera la determinación del tiempo que debe manejar el DTC o el componente COM+ expuesto?				
RESPUESTAS / ENCUESTADOS	A	B	C	D
1		X		
2		X		
3		X		
4	X			
5		X		
6		X		
7	X			
8		X		
9		X		
10		X		
11		X		
SUMATORIAS	2	9		

2. ¿Cómo considera que afectará la desactivación automática del componente expuesto luego de la llamada al método transaccional , no importando el resultado del mismo?				
RESPUESTAS / ENCUESTADOS	A	B	C	
1	X			
2	X			
3	X			
4	X			
5	X			
6	X			
7	X			
8	X			
9	X			
10	X			
11	X			
SUMATORIAS	11	0		

3. ¿Qué tan adecuada considera la implementación de la clase de fachada y su respectivo almacenamiento en las variables de aplicación a nivel del aplicativo participante?				
RESPUESTAS ENCUESTADOS	A	B		
1	X			
2	X			
3	X			
4	X			
5	X			
6	X			
7	X			
8	X			
9	X			
10	X			
11	X			
SUMATORIAS	11	0		

4. ¿Cómo considera las extensiones realizadas al protocolo WS-COORDINATION?				
RESPUESTAS ENCUESTADOS	A	B		
1	X			
2	X			
3	X			
4	X			
5	X			
6	X			
7	X			
8	X			
9	X			
10	X			
11	X			
SUMATORIAS	11	0		

5. ¿Cómo considera en líneas generales la Exposición del Modelo Transaccional COM+ para su Utilización En Actividades Transaccionales Todo o Nada Basadas en Servicios Web?.				
RESPUESTAS ENCUESTADOS	A	B	C	D
1	X			
2	X			
3		X		
4	X			
5	X			
6	X			
7	X			
8		X		
9	X			
10	X			
11		X		
SUMATORIAS	8	3		