Message Ordering Framework for Collaborative Web Services-based Environments

M. Vargas-Santiago, L. Morales-Rosales, S. Pomares-Hernandez, and K. Drira

***Abstract*—Web service paradigm is becoming a very powerful architecture for organizations when integrating heterogeneous applications. These provide functionality and form the basis for complex distributed business processes. Open standards make suitable Web Services interoperable for distributed environments. Collaboration between organizations is crucial in this context since it allows Web users to share knowledge, ideas, and modify information. Sharing information in a collaborative manner can minimize time spent in problem resolution. Message ordering is critical in this context, information presented to each user must be consistent way to preserve data integrity. For this purpose, causal ordering protocols are essential while exchanging information, however, their implementation is expensive to set up in distributed systems. Ongoing studies try to reduce the overhead imposed by the information carried out by each message, the optimal way of reducing such overhead is implementing the Immediate Dependency Relationship (IDR). In this paper, we present a framework for such message ordering, relative in collaborative environments, maintaining low overhead and computational cost based on the IDR.**

***Index Terms*—SOA, Web services, Message order, Collaborative, distributed systems, JMS.**

I. Introducción

LOS servicios Web están cambiando la forma en que vemos a los sistemas distribuidos, ya que estos proporcionan una arquitectura para integrar aplicaciones que se ejecutan en entornos distribuidos heterogéneos [13]; por lo tanto, estos pueden ser fácilmente integrados, por ejemplo, por un bus de servicios empresariales (ESB, Enterprise Service Bus). Hoy en día, los servicios Web son usualmente aplicaciones que se describen, publican y se accede a través de la Web usando estándares XML abiertos [1], [2], de aquí que ellos son las bases para la composición de procesos de negocios complejos [14]. La infraestructura de los servicios Web proporciona los fundamentos para construir un protocolo de intercambio flexible y extenso [3]. No obstante, las aplicaciones de los clientes suelen utilizar el protocolo HTTP como forma de conexión al invocar a los servicios Web; sin embargo, HTTP no garantiza el ordenamiento de mensajes dentro de los entornos colaborativos, además de que no admite el intercambio de mensajes asíncronos; por lo cual es necesario un mecanismo de mensajería más robusto. Como consecuencia, los servicios Web pueden configurarse para que las aplicaciones cliente también puedan utilizar el servicio de mensajes Java (JMS, Java Message Service) como su mecanismo de transporte.

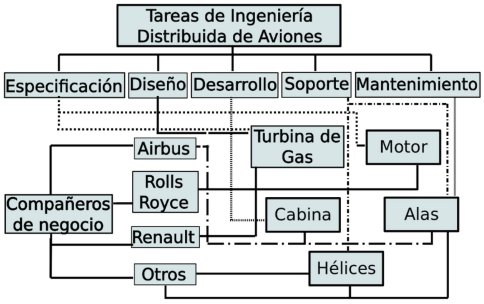


Fig. 1. Ambiente Colaborativo para el diseño de Aeronaves.

JMS se puede configurar en dos estilos de comunicación diferentes basados en mensajes: punto a punto (P2P) y publicar/suscribirse. En el estilo P2P, cada mensaje se envía a una cola específica desde donde el receptor extrae sus mensajes. En el estilo de publicación /suscripción tanto los editores como los suscriptores publican o suscriben dinámicamente la jerarquía de contenido. Debido a la simplicidad de JMS, se ha convertido en una de las soluciones más utilizadas para desarrollar aplicaciones colaborativas escalables.

Las soluciones y entornos colaborativos permiten a los usuarios modificar y compartir conocimientos, ideas e información entre sí de manera eficaz. Éste tipo de soluciones se han vuelto muy populares entre las organizaciones porque dan mayor agilidad, minimizan los esfuerzos y reducen el tiempo dedicado a la resolución de problemas, dando una mejor sinergia entre las organizaciones y aumentando así la eficiencia y eficacia de sus colaboraciones [4], [5], [6]. Los entornos de misión crítica basados en la Web han sido objeto de estudio; en muchos casos los servicios Web se utilizan para descubrir una funcionalidad de negocio, presentada como servicio estando disponibles a través de la red y son compartidos e invocados por diversos socios corporativos. El compartir y descubrir información en un contexto colaborativo es exigido por el desarrollo industrial de redes dinámicas. En los escenarios colaborativos distribuidos, los socios comerciales deben compartir y modificar la información de forma remota. Por ejemplo, considere la Fig. 1 (ésta muestra el ciclo de vida del producto) donde un conjunto de socios de aeronaves necesitan desarrollar, diseñar, dar mantenimiento, apoyar, y especificar

M. Vargas was with the Department of Computer Science, National Institute for Astrophysics, Optics and Electronic (INAOE) , Tonantzlintla, Puebla, Mexico e-mail: mariano.v.santiago@ccc.inaoep.mx

L. Morales-Rosales was with the Faculty of Civil Engineering, Universidad Michoacana de San Nicolas de Hidalgo, Morelia, Michoacan, Mexico

S. Pomares-Hernández was with the Department of Computer Science, National Institute for Astrophysics, Optics and Electronic (INAOE) and CNRS, LAAS, 7 avenue du Colonel Roche, Toulouse, France and Univ de Toulouse, LAAS, Toulouse, France

K. Drira was with LAAS, 7 avenue du Colonel Roche, Toulouse, France and Univ de Toulouse, LAAS, Toulouse, France

los componentes de la aeronave, tales como: turbinas de gas, motores, cabina, hélices y alas. Estos procesos colaborativos pueden ser representados como servicios Web y exponer sus sistemas de Diseño Asistido por Computadora (CAD, Computer-aided Design) como tales.

Asegurar el orden de los mensajes dentro de los entornos colaborativos es fundamental ya que todos los usuarios involucrados deben tener la misma visión del sistema y los datos deben conservar su integridad; además, los mensajes proporcionan el comportamiento esperado por las aplicaciones distribuidas. Para este propósito, los protocolos de ordenamiento causal son esenciales para el intercambio de información, sin embargo su implementación es costosa de implementar en los sistemas distribuidos [7]. La forma óptima de disminuir dicha sobrecarga es mediante la implementación de la Relación de Dependencia Inmediata (IDR, Immediate Dependency Relationship). De hecho, la IDR puede asegurar la entrega causal global de mensajes en la comunicación en grupo y anula la noción de que la causalidad puede ser costosa de implementar en sistemas distribuidos; reduce considerablemente la cantidad de información de control, información contenida en cada mensaje para preservar la integridad de los datos [8].

En este trabajo se propone un framework para el ordenamiento de mensajes (MOF, Message Ordering Framework) para entornos colaborativos; este asegura el ordenamiento causal de acuerdo con la visión causal de los sistemas involucrados, manteniendo un bajo costo de sobrecarga y bajo costo computacional ya que se basa en la IDR.

El trabajo está organizado de la siguiente manera: la Sección II presenta brevemente los antecedentes. La Sección III da los trabajos relacionados. La Sección IV describe el enfoque propuesto (MOF) e ilustra un caso de uso. Los resultados experimentales se muestran en la Sección V. Finalmente, las conclusiones y trabajo futuro se discuten en la Sección VI.

II. Antecedentes

A. Modelo del Sistema

Los sistemas distribuidos están caracterizados por la carencia de un reloj global, es decir, no existe una noción global del tiempo. Además, los participantes o procesos no comparten memoria, y se comunican exclusivamente mediante el intercambio de mensajes. Es decir existen tres componentes principales:

- **Procesos**. Los participantes, en nuestro caso servicios Web, se representan por procesos individuales. Cada proceso pertenece al conjunto $P = p_i, p_j, \ldots$ y se comunican intercambiando mensajes con otros procesos, enviando únicamente un mensaje a la vez.
- **Mensajes**. Estos son los mensajes intercambiados en el ambiente colaborativo y pertenecen al conjunto M. Par nuestro enfoque $m \in M$ está caracterizado por $(i, t_i, mensaje, H_m)$ donde:
  - i es el identificador del proceso que origino el mensaje.
  - t_i es el tiempo lógico del proceso p_i. El valor de t_i es manejado mediante la conocida estructura de *Reloj Vectorial*.
  - $mensaje$ contiene la información relativa a la aplicación.
  - H_m contiene la información de control que guarda la IDR entre participantes.
- **Eventos**. Nosotros consideramos tres clases de eventos:
  - *Envío (send)* se refiere a la emisión de un mensaje ejecutado por cualquier proceso.
  - *Recepción (receive)* se refiere a la notificación de llegada de un mensaje en un proceso.
  - *Entrega (delivery)* se refiere a la ejecución realizada por un proceso para presentar la información recibida a una aplicación u otro proceso.

  Sea m un mensaje. Denotamos por $send(m)$ el evento de emisión, por $receive(m)$ a la recepción, y por $delivery(p, m)$ a la entrega de m al proceso p. El conjunto de eventos asociados a M es:
  $E_m = send(m) : m \in M \cup delivery(p, m) : m \in M \forall P \in P$ Así todo el conjunto de eventos es:
  $E = E_i \cup E_m$

B. Definiciones

La Relación Sucedió Antes (HBR, Happened Before Relation). Esta relación establece dependencias de precedencia causales sobre un conjunto de eventos. La HBR es un estricto orden parcial (transitivo, irreflexiva y anti-simétrica). Se denotada por $\rightarrow$, y es definida por las siguientes tres reglas [9]:

1) Definición 1:

1) Si a y b son eventos del mismo proceso y a fue originada antes que b, entonces $a \rightarrow b$.
2) Si a es el evento de envío $send(m)$ y b es el evento de entrega $delivery(p, m)$, entonces $a \rightarrow$.
3) Si $a \rightarrow b$ y $b \rightarrow c$, entonces $a \rightarrow c$.

La Relación de Dependencia Inmediata (IDR, Immediate Dependency Relation). La HBR en la práctica es cara de implementar ya que tiene que hacer un seguimiento de la relación entre cada par de eventos. Con el fin de evitar la costosa implementación en práctica, la IDR identifica y adjunta la cantidad mínima de información de control por mensaje para asegurar el orden causal del sistema. La IDR es la reducción transitiva de la HBR, y se denota por "$\downarrow$", y se define de la siguiente manera [8]:

2) Definición 2: Dos mensajes a y $b \in E$ tienen una IDR $a \downarrow b$ si se satisface la siguiente relación:

$$a \downarrow b \iff [a \rightarrow b \wedge \forall c \in E, \neg(a \rightarrow c \rightarrow c)]$$

III. Trabajos Relacionados

En [5] Wu et. al., proponen un sistema colaborativo, un framework basado en servicios Web, en particular implementan su solución para el control de video conferencias y por consiguiente integrando diversas tecnologías; con el fin de controlar las colaboraciones de audio y video multipunto. En otras palabras, es una forma sofisticada de integrar aplicaciones colaborativas como H.323, SIP y Access Grid en un solo entorno. Sin embargo, no tienen en cuenta que los mensajes deben presentarse adecuadamente a los usuarios finales, ya que deben tener una representación coherente de los datos.

En [6], los autores sugieren un framework para la integración de tecnologías heterogéneas; específicamente herramientas de colaboración que tienen la necesidad de interactuar. En este trabajo, Oliva et. al., buscan establecer un enfoque estandarizado, para dicho propósito utilizan REST, Representational State Transfer. Un estilo arquitectónico que especifica las restricciones aplicadas a los servicios Web con los cuales se inducen propiedades deseables, como el rendimiento y la escalabilidad. Los servicios Web REST tienen como objetivo integrar diferentes modelos de datos, motores de flujo de trabajo o reglas de negocio. Sin embargo, al utilizar REST, las aplicaciones se ejecutan en la World Wide Web mediante el protocolo HTTP para transferir datos, por lo que se necesita un ordenamiento de mensajes más robusto para preservar la coherencia de los datos.

En [4] Pomares et. al., proponen un protocolo de coordinación para las actividades de ingeniería de colaboración, evitando escenarios de colaboración errónea en componentes distribuidos y aplicaciones. Aunque este trabajo no está basado en servicios Web, tiene como principal objetivo asegurar el ordenamiento causal de mensajes específicamente implementando la IDR y de esta forma reduciendo la sobrecarga transmitida por cada participante. Sin embargo, los servicios Web proporcionan interoperabilidad para entornos colaborativos complejos.

La Tabla I resume los trabajos relacionados, expone el objetivo, la tecnología utilizada y el entorno en el que los autores proponen sus soluciones. A pesar de las diferentes propuestas que provienen de los trabajos relacionados, aún quedan pendientes algunas preguntas como: ¿Cómo integrar a los servicios Web en entornos dinámicos de forma autónoma sin perder el orden de los mensajes y como mantener información congruente?

En la actualidad existen otros enfoques que todavía se proponen para el ordenamiento de mensajes [10], [11], [12]. Por ejemplo, en [11] el autor hace uso de otros componentes de software tales como un intermediario (middleware) de mensajes que temporalmente almacena mensajes y los procesa en el orden en que fueron recibidos. En [10] el autor presenta una forma de ordenar mensajes mediante el uso de etiquetas y proxies, sin embargo ambos extremos deben estar de acuerdo en la comunicación tanto en la secuencia y la etiqueta inicial. De tal manera que es entonces que el proxy lee esta etiqueta y reordena los mensajes si llegan fuera de orden.

Por otro lado, en la industria también se aborda el problema de ordenamiento de mensajes. Por ejemplo, Oracle® [12] propone un ordenamiento estricto de mensajes, para ello utiliza el servidor JMS WebLogic. Pero este es un software propio con el cual ellos logran distinguirse de otros métodos, es decir, es su valor agregado. Para lograr garantizar el orden de los mensajes con respecto al orden de procesamiento de un grupo de mensajes; estos se almacenan agrupándolos en una sola unidad llamada Unidad de ordenamiento.

TABLA I
TRABAJOS RELACIONADOS

Parámetro	[4]	[5]	[6]
Objetivo	Evitar escenarios erróneos	Integración de múltiples sistemas de conferencia	Integración de múltiples sistemas de conferencia
Tecnología usada	Causalidad e IDR	Servicios Web	Servicios Web REST
Ambiente	Ingeniería distribuida	Video conferencias	Herramientas heterogéneas de trabajo

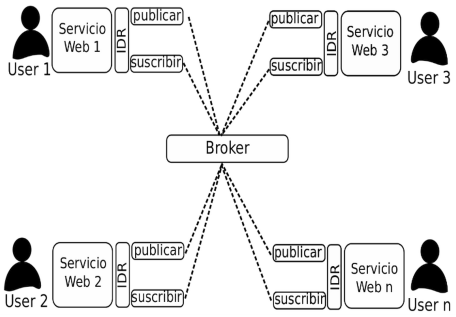


Fig. 2. Entorno colaborativo utilizando MOF.

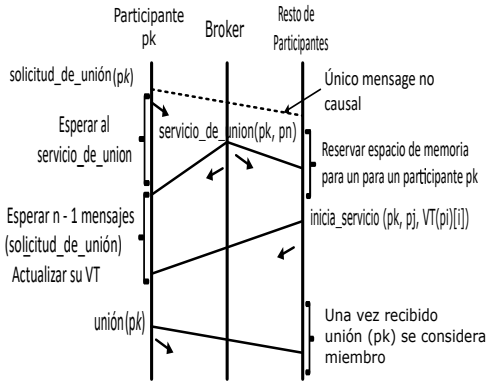
IV. MOF PARA ENTORNOS COLABORATIVOS BASADOS EN SERVICIOS WEB

Los entornos colaborativos de misión crítica necesitan un mecanismo de transporte confiable y robusto, particularmente cuando el orden del mensaje es importante, pero JMS por sí solo no puede conceder tales garantías en una comunicación en grupo. El marco de referencia (framework) para el ordenamiento de mensajes (MOF) se basa en JMS; pero se le han ampliado sus propiedades. Es decir, todas las comunicaciones que pasan a través de él se enriquecen con la pequeña sobrecarga, que consiste en información de control, para el seguimiento en el ordenamiento de mensajes. MOF utiliza un middleware de mensajería, es decir, un intermediario (bróker), para soportar redes heterogéneas distribuidas y para dar soporte al modelo de comunicación publicación/suscripción. Donde se vinculan de forma autónoma a varios publicadores y suscriptores. Otra característica noble de MOF es que no depende de los servicios Web en lugar.

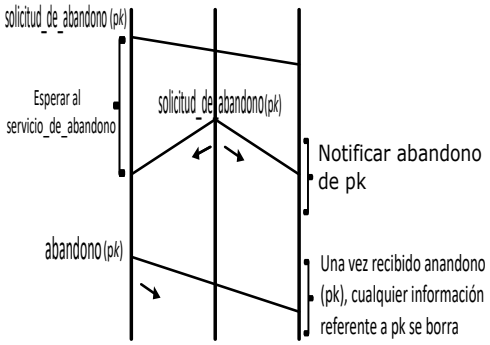
Con dicho propósito, presentamos el esquema mostrado en la Fig. 2 donde el intermediario está a cargo del servicio de membresía de los usuarios, en otras palabras de cómo se suscriben y salen automáticamente mientras se mantienen las propiedades de causalidad entre los usuarios que utilizan al sistema. Con este fin se muestra la Fig. 3a la suscripción y en la Fig. 3b la salida, de los usuarios del entorno colaborativo.

A. Suscripción

La Fig. 3a muestra la suscripción de un nuevo usuario al entorno de trabajo colaborativo. Cuando un participante o usuario quiere unirse, debe enviar una solicitud de admisión al bróker o intermediario el cual les permite a todos los demás usuarios saber que un nuevo usuario se está uniendo con el mensaje ($solicitud\_de\_union(p_k)$), siendo éste el único mensaje no causal. Después, el nuevo usuario debe esperar hasta que se reciba un mensaje de unión



(a) Servicio de suscripción



(b) Servicio de salida

Fig. 3. Servicio de membresía en un entorno colaborativo.

al servicio $servicio\_de\_union(p_k, p_n)$. Una vez recibido, el nuevo usuario debe esperar la fase de inicialización $inicia\_servicio(p_k, p_i, VT(p_i)[i]))$ donde se le asigna su valor de reloj vectorial. Finalmente, el usuario debe enviar un mensaje de unión al bróker $union(p_k)$ que lo envía a todos los demás usuarios. Una vez recibido, el usuario está debidamente integrado al entorno colaborativo.

B. Salida de la Membresía

La Fig. 3b muestra la salida de un usuario del entorno de trabajo colaborativo. Cuando un usuario quiere salir, primero debe enviar una solicitud de petición abandono $solicitud\_de\_abandono(p_k)$, que es enviada por el bróker a todos los demás usuarios anunciando que un usuario abandonará el entorno colaborativo. Finalmente, el usuario que se va, envía el mensaje $abandono(p_k)$ al bróker; una vez recibida por todos los demás participantes el usuario tiene una salida adecuada del entorno de colaboración y, por tanto, la causalidad se conserva entre el resto de los participantes.

C. Arquitectura de MOF

El objetivo del MOF es proponer un framework extensible para el API JMS que pueda ser explotado o aprovechado por los servicios Web, particularmente en entornos de trabajo colaborativo sobre redes heterogéneas. Este se basa en la IDR y tiene una sobrecarga baja y mantiene la coherencia de la información. MOF es óptimo porque transmite la cantidad mínima y necesaria de información de control para preservar completamente el orden causal entre mensajes o eventos; También es capaz de gestionar la interoperabilidad y escalabilidad porque está construido para los servicios y el IDR está diseñado para hacer frente a grandes sistemas distribuidos.

La Fig. 4 ilustra en detalle las comunicaciones que tienen lugar en un entorno de trabajo colaborativo, en el que tiene lugar la publicación (proveedores del servicio) en una cola o en un tópico y la suscripción (consumidores del servicio) donde se puede recuperar información colaborativa. En cualquier forma de comunicación ya sea de publicación o suscripción la comunicación pasa a través del bróker para posteriormente ser entregada al MOF.

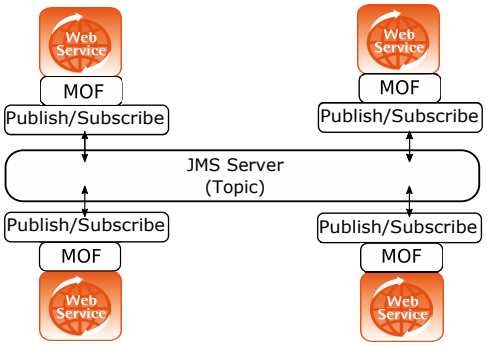


Fig. 4. Entorno colaborativo.

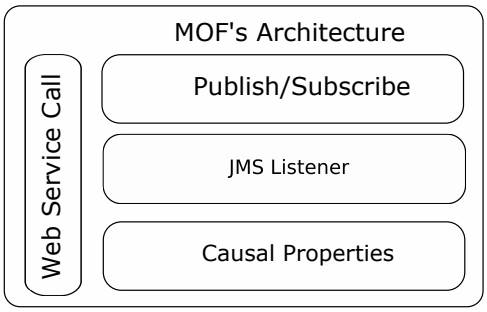


Fig. 5. Arquitectura de MOF.

La Fig. 5 muestra la arquitectura del framework MOF. Los servicios Web subyacente son descubiertos y administrados por el bróker, el cual despliega a los servicios Web como mensajes al MOF. Las principales acciones del resto de los componentes son:

- *Llamada a servicio Web* es el componente que se encarga de la comunicación con los servicios Web, ya que usualmente dichos están alojados en servidores remotos; es necesario un mecanismo que interprete a los stubs y callback handlers proporcionados por terceros.
- El componente *Publicar/Suscribir* es el encargado de publicar o leer el contenido, dependiendo del tipo de mensaje.
- El *JMS Listener*, es el encargado de escuchar a todos los eventos o mensajes intercambiados por los diversos

usuarios. También, de la comunicación con los *stubs* y los *callback handler*; es decir, traduce la información a un lenguaje que entienden tanto los servicios Web como el componente *Propiedades Causales*.

- Finalmente, el componente *Propiedades Causales* se encarga de mantener el orden de los mensajes. Para lograr esto, las peticiones o respuestas se enriquecen con la IDR. Para mantener nuestro framework óptimo, la información causal con estampilla de tiempo por mensaje corresponde a los mensajes conectados por una IDR.

D. Algoritmo IDR

Algoritmo 1 Relación de dependencia inmediata

procedure INICIALIZACIÓN() ▷ Para cada servicio Web la información de control se vacía
$VT(p_i)[j] = 0 \; \forall j : 1, 2, ..., n$
$CI_i \leftarrow \emptyset$
end procedure
procedure DIFUSIÓN(m)
$VT(p_i)[i] = VT(p_i)[i] + 1$ ▷ Se actualiza reloj
$H_m = CI_i$ ▷ Se actualiza la información de control
$m = (i, t_i = VT(p_i)[i], message, H_m)$ ▷ Se da formato al mensaje a enviar.
Send(m) ▷ Se difunde el mensaje
$CI_i \leftarrow \emptyset$ ▷ Se actualiza información de control
end procedure
procedure RECEPCIÓN(m) for $p_j, i \neq j$
$m = (k, t_k, message, H_m)$ ▷ Se verifica la condición de entrega. Para asegurar la entrega causal del mensaje *m*
if not$(t_k = VT(p_j)[k] + 1$**and**$t_l \leq VT(p_j)[l] \forall (l, t_l) \in H_m)$ **then**
encola(m) ▷ Se encola el mensaje que no cumple la condición de entrega
else
entrega (m)
$VT(p_j)[k] = VT(p_j)[k] + 1$ ▷ Se actualiza la entrada del reloj correspondiente en el servicio Web receptor
if $\exists ci_{s,t'} \in CI_j | k = s$ **then**
$CI_j \leftarrow CI_j \backslash \{ci_{s,t'}\}$ ▷ Se borran los valores antiguos de la información de control
end if
$CI_j \leftarrow CI_j \cup \{(k, t_k)\}$
$CI_j \leftarrow CI_j \backslash H_m$ ▷ Se actualiza la información de control
end if
end procedure

El algoritmo de la IDR se divide en dos partes [8]; la primera parte es para el envío de mensajes o difusión y la segunda parte es para la recepción o de entrega causal de los mensajes. El Algoritmo 1, muestra las dos partes, cabe mencionar que este algoritmo es ejecutado por cada uno de los participantes o procesos (servicios Web) del entorno colaborativo.

V. RESULTADOS EXPERIMENTALES

Con el objetivo de demostrar que nuestro framework no tiene gran impacto en el rendimiento general de los sistemas, realizamos pruebas de rendimiento. En concreto, medimos el tiempo de respuesta como indicador clave de rendimiento. Para las pruebas de rendimiento implementamos nuestro framework dentro del siguiente hardware: en una estación de trabajo (workstation) con 16 GB de RAM con Windows 7 64 bits como sistema operativo. Se utilizaron dos middlewares o brókers (Jboss versión 5.1 y Glassfish versión 4.0), primero se utilizó el servidor WSO2 Application Server para desplegar a los servicios Web, de manera similar se utilizó Glassfish para desplegar a los servicios Web, y para las pruebas de rendimiento se emularon diversos clientes en Java. Tomamos como escenario de prueba diferentes servicios Web capaces de interactuar, con dichos emulamos un ambiente de trabajo colaborativo (ver Fig. 4).

Las figuras Fig. 6a y Fig. 6b muestran de manera cuantitativa la variación que presenta el tiempo de respuesta. Este va aumentado de acuerdo al número de procesos involucrados, esto es de esperarse en el modo broadcast (difusión) ya que se está evaluando el peor caso, cuando todos los participantes transmiten al mismo tiempo; en teoría la probabilidad de que esto suceda es muy baja cuando se tienen bastantes procesos. Este es un problema del entorno en sobre el cual se está trabajando y no del enfoque propuesto. En nuestra propuesta el tiempo de respuesta se mantiene casi constante aun cuando el número de procesos en el sistema incrementa.

De estas dos figuras (Fig. 6a y Fig. 6b), se puede argumentar que nuestro enfoque es escalable y con bajo costo de implementación (en cuanto al impacto en el rendimiento). Ya que para el tiempo de respuesta no se violan los valores recomendados por la UTI G.1010, el cual intenta estandarizar el uso de servicios Web. Estipula los siguientes valores: *preferido = 0-2 segundos, aceptable = 2-4 segundos y e inaceptables = 4 a infinito*.

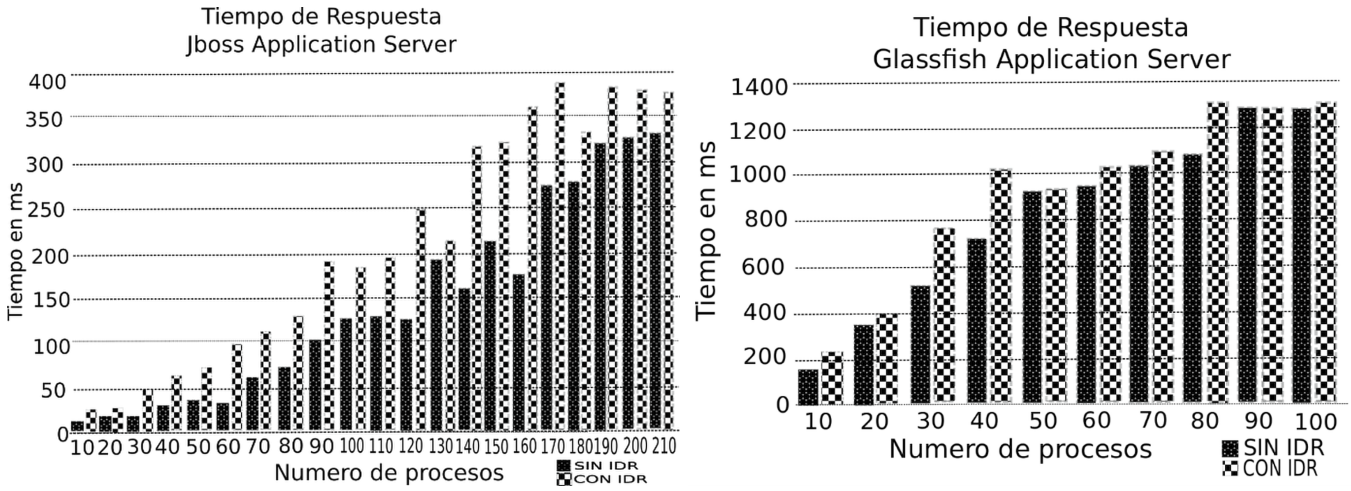
De las pruebas de rendimiento podemos concluir que el rendimiento del sistema no se afecta cuando se implementa el *framework* propuesto para el ordenamiento de mensajes. La Tabla II muestra el porcentaje que incrementa nuestro enfoque, este porcentaje fue medido tanto para las pruebas de rendimiento realizadas con el bróker *Jboss* como para el bróker *Glassfish*.

De la Tabla II, se puede concluir que parecería que nuestro enfoque incrementa demasiado el uso de recursos al emplearlo con pocos procesos, y que afecta al sistema, pero en realidad nuestro enfoque es escalable ya que a mayor número de procesos o participantes en el sistema menor se ve afectado el rendimiento del mismo.

En la Tabla II sólo se muestran hasta 100 usuarios transmitiendo concurrentemente al servidor Glassfish, ya que presentan el mismo comportamiento utilizando y sin utilizar nuestro *framework*.

VI. CONCLUSIÓN

En este trabajo mostramos que es de vital importancia respetar el orden bajo el cual se ejecutan o llevan a cabo



(a) Utilizando el servidor *JBoss*

(b) utilizando el servidor *Glassfish*

Fig. 6. Medición de tiempo de respuesta del sistema implementado la IDR y sin su implementación.

TABLA II
PORCENTAJE INCREMENTADO

Procesos	Incremento (%) Usando Jboss	Incremento (%) Usando Glassfish
10	100	26.13636364
20	108.3333333	6.382978723
30	163.1578947	42.26618705
40	90.90909091	42.81729428
50	66.66666667	0.760869565
60	163.1578947	9.768907563
70	81.96721311	8.882783883
80	76.62337662	16.6380789
90	88.23529412	0.665680473
100	35.8778626	0.896860987
110	40.14084507	
120	53.28467153	
130	28.86597938	
140	105.8064516	
150	49.31506849	
160	103.4285714	
170	38.18181818	
180	21.14695341	
190	21.25	
200	16.36363636	
210	13.52941176	

funciones o métodos que muchas de las veces son compartidas por diversas organizaciones. La información debe respetar el orden de ejecución ya que está en los sistemas distribuidos representa el comportamiento del sistema.

Uno de los pioneros en el ordenamiento de mensajes fue Leslie Lamport, el propone realizar un orden estricto entre pares de mensajes intercambiados por distintos procesos. Pero su relación, la relación sucedió antes HBR (*Happened-before relation*) es costosa de implementar en la práctica. Por lo tanto surgen diversos enfoques, pero muchas veces son software propio de la empresa y algunas veces no siempre siguen estándares.

Nuestro enfoque sigue estándares, está basado sobre JMS, es interoperable está orientado a los ambientes colaborativos basados en servicios Web, es decir, se usa la interfaz de los servicios Web para soportar las colaboraciones entre distintas organizaciones. Además, de las pruebas de rendimiento se concluye que nuestro enfoque es escalable ya que se ilustra un comportamiento que no impacta de manera negativa al sistema aun cuando el número de procesos involucrados incrementa de gran manera. En otras palabras la IDR se vuelve aceptable o viable en su implementación ya que a mayor número de procesos el rendimiento del sistema es escalable como se mostró en las pruebas de rendimiento.

AGRADECIMIENTOS

Este trabajo es financiado por el Consejo Nacional de Ciencia y Tecnologia de Mexico (CONACYT) a través del proyecto ID PDCPN2013-01-215421.

REFERENCIAS

[1] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana, *Unraveling the web services web: an introduction to SOAP servicies, WSDL, and UDDI* IEEE Internet computing 6 (2) (2002) 86-93.

[2] S. Tai, T. A. Mikalsen, I. Rouvellou, *Using message-oriented middleware for reliable Web services messaging* In International Workshop on Web Services, E-Business, and the Semantic Web (pp. 89-104). Springer, Berlin, Heidelberg.

[3] L.-J. Zhang, M. Jeckle, *The next big thing: Web services collaboration*, In Web Services-ICWS-Europe 2003 (pp. 1-10). Springer, Berlin, Heidelberg.

[4] S.-E.-Pomares, Drira,-K., Fanchon,-J., Diaz,-M. *An efficient multi-channel distributed coordination protocol for collaborative engineering activities.* In IEEE International Conference on Systems Man and Cybernetics (SMC'02) (p. 6p).

[5] W. Wu, G. C. Fox, H. Bulut, A. Uyar, H. Altay, *Design and implementation of a collaboration Web-services system*.

[6] L. Oliva, L. Ceccaroni, *REST Web services in collaborative work environments*, In CCIA (pp. 419-427).

[7] Kshemkalyani, Ajay D., and Mukesh Singhal. *Distributed computing: principles, algorithms, and systems*. Cambridge University Press, 2011.

[8] S. Pomares, J. Fanchon, and K. Drira. *The immediate dependency relation: an optimal way to ensure causal group communication*. In Annual Review of Scalable Computing, pp. 61-79. 2004.

[9] L. Lamport, *Time, clocks, and the ordering of events in a distributed system*, Communications of the ACM 21 (7) (1978) 558-565.

[10] Schumacher, D. (2011, July 15). *In-Order Message Delivery*. Retrieved March 28, 2016, from http://www.dalnefre.com/wp/2011/07/in-order-message-delivery/, http://www.dalnefre.com/wp/2011/07/in-order-message-delivery/In-Order Message Delivery

[11] Chanaka, F. (2013, December 09). *Building an In-Order, Guaranteed Delivery Messaging System for Your Enterprise with WSO2 Products*. Retrieved March 28, 2016, from goo.gl/pf8hac, https://wso2.com/library/articles/2013/12/building-an-in-order-guaranteed-delivery-messaging-system-for-your-enterprise-with-wso2-products/Guaranteed Delivery Messasing System

[12] Oracle®. (2014, February 12). *Fusion Middleware Programming JMS for Oracle WebLogic Server.* Retrieved March 28, 2016, from https://docs.oracle.com/cd/E24329_01/web.1211/e24387/toc.htm, https://docs.oracle.com/cd/E24329_01/web.1211/e24387/toc.htmOracle WebLogic Server

[13] Santiago, M. V., Hernandez, S. E. P., Rosales, L. A. M., Kacem, H. H., *Fault tolerance approach based on checkpointing towards dependable business processes*. IEEE Latin America Transactions, (2016), 14(3), 1408-1415.

[14] Scott, E., Soria, A., Campo, M. *A Taxonomy-based Approach For Fault Localization In Service-Oriented Applications*. IEEE Latin America Transactions, (2016), 14(5), 2348-2354.

Mariano Vargas-Santiago received the Master in Research degree in Information and Telecommunications from l'Institut National des Sciences Appliquées de Toulouse, Toulouse, France, in 2013. Presently he is a PhD candidate in Distributed Systems and Software Engineering at El Instituto Nacional de Astrofísica, Óptica y Electrónica, Mexico, where his current research interests include the merging of two widely used paradigms: checkpointing mechanisms and autonomic computing.

Luis Morales-Rosales received an Engineering Degree in Computer Systems from the Instituto Tecnológico de Colima, Colima, Mexico in 2001 and his PhD in Computer Science in 2009 from El Instituto Nacional de Astrofísica, Óptica y Electrónica, Mexico. His current research interests are Applications of Distributed Systems and Intelligent Computing.

Saúl Pomares-Hernández completed his PhD Degree at the Laboratory for Analysis and Architecture of Systems of Le Centre National de la Recherche Scientifique, France, 2002. Currently he is a researcher (Professional Title) in the Computer Science Department at El Instituto Nacional de Astrofísica, Óptica y Electrónica, Mexico. Since 1998 he has been conducting research in the field of Distributed Systems, Partial Order Algorithms and Multimedia Synchronization.

Khalil Drira received Engineering and M.S. (DEA) Degrees in Computer Science from The École Nationale Supérieure d'Electrotechnique, d'Electronique, d'Informatique, d'Hydraulique et des Télécommunications (ENSEEIHT), Toulouse, in 1988. He obtained PhD and HDR degrees in Computer Science from l'Université Paul Sabatier Toulouse in 1992 and 2005 respectively. Since 1992 as Charg de Rechere, he assumes a full time research position in Le Centre National de la Recherche Scientifique, France. His research activity addresses topics in this field and he continues to be involved in national and international projects. Khalil continues as a member of the program journals in the fields of software architecture as well as communicating. He has also been an editor of a number of proceedings, books and journals which have been published and presented in these fields.