Abstract—Web Services paradigm is becoming a very powerful architecture for organizations in integrating heterogeneous applications; these provide functionality and form the basis for complex distributed business processes. Open standards make suitable Web Services inter-operable for distributed environments. Collaboration between organizations is crucial in this context since it allows 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 must be presented to each user in a 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 it is based on the IDR.

Index Terms—Collaborative Web Services; JMS; message ordering;

I. INTRODUCTION

Web Services (WS) are changing the way we see distributed systems since they provide an architecture for integrating applications running in heterogeneous distributed environments; therefore these can be easily integrated, for example by an Enterprise Service Bus (ESB). Today, WS are usually applications that describe, publish and are accessed over the Web using open XML standards [13] [12], thus, WS are the base compositions for complex business processes. WS infrastructure provide a good foundation to build a flexible and extensive exchange protocol [14]. However, client applications usually use the HTTP as connection protocol when invoking WS. But, HTTP does not guarantee message ordering delivery in collaborative environments, plus it does not support asynchronous messages exchange; therefore, a more robust messaging mechanism is needed. In this manner, WS can be configured so that client applications can also use the Java Message Service (JMS) as their transport mechanism. JMS can be configured in two different message-based communication styles: point-to-point(P2P) and publish/subscribe. In the P2P style each message is sent to a specific queue from where the receiver extracts their messages. In the publish/subscribe style both publishers and subscribers dynamically publish or subscribe to the content hierarchy. Because of JMS’s simplicity, it has become one of the most used solutions for developing scalable collaborative applications.

Collaborative environments and solutions allow users to modify and share knowledge, ideas and information among each other effectively. They have become very popular among organizations because they give greater agility, minimize duplicate efforts and reduce time spent in resolution of issues, giving a better synergy between organizations and thereby increasing the effectiveness and efficiency of their collaborations [4] [13] [9]. Web-based mission critical environments have been a subject of study; in many cases WS are used to discover a business functionality, presented as services, that are available through the network and are shared and invoked by corporate partners. Sharing and discovering information in a collaborative context is demanded by the industrial development of dynamic networks. In distributed collaborative scenarios business partners need to share and modify information remotely. For example, consider Fig. 1 (showing a product life cycle); a set of aircraft partners need to maintain, support, develop, design and specify aircraft components: gas turbines, engines, cabin, propellers, and wings. These collaborative processes can be represented as Web services and expose their Computer-aided Design (CAD) systems as such.
Assuring message ordering for collaborative environments is fundamental since all the involved users should have the same view of the system and data must retain its integrity; additionally, the messages provide the expected behavior by the distributed applications. For this purpose, causal ordering protocols are essential for exchanging information, however their implementation is expensive to set up in distributed systems [7]. The optimal way of diminishing such overhead is by implementing the Immediate Dependency Relationship (IDR). Indeed, the IDR can ensure global causal delivery of messages in group communication and it obliterates the notion that causality can be expensive to implement in distributed systems; it considerably reduces the amount of control information; information carried in each message to preserve data integrity [5].

In this work we propose a Message Ordering Framework (MOF) for collaborative environments; this ensures causal ordering according to the causal view of the systems involved while maintaining a low overhead and computational cost since it is based on the IDR.

The paper is organized as follows: Section 2 introduces briefly the causality principle and the background of IDR. Section 3 describes related works. Section 4 presents the implementation of the Message Ordering Framework (MOF). Conclusions and future work are provided in Section 5.

II. BACKGROUND

Leslie Lamport defined the happened-before relation (HBR) trying to totally order events in distributed systems [3]: the HBR is also known as the causal precedence relation. It orders messages by pairs using identifier id and scalar time in the following manner:

**Definition 1:** The causal relation, denoted by $\rightarrow$, is defined by the following three rules [5]:

1) $\langle i, a \rangle \rightarrow \langle j, b \rangle$ if $i=j \land a < b$.
2) $\langle i, a \rangle \rightarrow \langle j, b \rangle$ if $\langle i, a \rangle$ is the sending of a message and $\langle j, b \rangle$ is the delivery of that message.
3) $\langle i, a \rangle \rightarrow \langle j, b \rangle$ if $\exists (k,c)(\langle i, a \rangle \rightarrow \langle k, c \rangle \land \langle k, c \rangle \rightarrow \langle j, b \rangle)$ where $i,j$ and $k$ are process identifiers, and $a,b$ and $c$ are local clock values of $i,j$ and $k$.

Causal ordering delivery in group communication exhibits two cases: the broadcast case and the multi-group case that includes overlapping groups. The first case is defined as follows [5].

**Definition 2:** Causal broadcast delivery, one group:

If $\text{send}(m) \rightarrow \text{send}(m')$, then $\forall k \in c$:

$\text{delivery}_k(m) \rightarrow \text{delivery}_k(m')$

Causal broadcast delivery stipulates that if the diffusion of a message $m$ causally precedes the diffusion of a message $m'$, in a group $c$, then the delivery of $m$ causally precedes the delivery of $m'$ for all participant $p_k$ that belong to $c$.

The second case is defined as follows:

**Definition 3:** Causal multi-group delivery [5]:

If $\text{send}_i(m, c) \rightarrow \text{send}_j(m', c')$, then $\forall k \in c \cap c'$:

$\text{delivery}_k(m) \rightarrow \text{delivery}_k(m')$

Causal multi-group delivery guarantees that if the diffusion of a message $(m, c)$ causally precedes the diffusion of a message $(m', c')$, then the delivery of $m$ causally precedes the delivery of $m'$ for all the participants $p_k$ that belong to the intersection of groups $c$ and $c'$.

**The Immediate Dependency Relation.** The HBR in practice is expensive since it has to keep track of the relation between each pair of events. In order to avoid such the Immediate Dependency Relation (IDR) identifies and attaches the minimal amount of control information per message to ensure causal ordering. The IDR is the transitive reduction of the HBR, and it is denoted by $\perp$, defined as follows:

**Definition 4:** Two messages $m$ and $m' \in M$ have an IDR $m \perp m'$ if the following restriction is satisfied:

$m \perp m' \leftrightarrow [m \rightarrow m' \land \forall m'' \in M, \neg (m \rightarrow m'' \rightarrow m')]$

III. RELATED WORKS

In [13] the authors propose a collaborative system, a framework based on WS, in particular they implement their solution for conference control integrating various technologies; controlling multipoint audio and video collaborations. In other words, it is a sophisticated way of integrating collaborative applications like H.323, SIP and Access Grid into a single environment. However, they do not take into account that messages need to be properly presented to the end users, since they must have a coherent representation of the data.

Another work which attacks collaborative work environments is exhibited in [9], the authors suggest a framework for the integration of heterogeneous technologies specifically collaborative tools which have the necessity of interacting. Also they want to establish a commonly standardized approach, using Representational State Transfer (REST); an architectural style that specifies constraints applied to WS inducing desirable properties, such as performance and scalability. REST WS aimed at integrating different data models, workflow engines or business rules. However, since the authors use REST, applications run in the World Wide Web using HTTP protocol to transfer data, thus a more robust message ordering is needed for preserving data coherence.

In [4] the authors propose a coordination protocol for collaborative engineering activities while avoiding erroneous collaboration scenarios in distributed components and applications. Although this work is not based on WS, it is based on causal message ordering, specifically in the IDR reducing the overhead transmitted by each participant. However, WS can provide the interoperability for complex collaborative environments.

Table I summarizes the related works. It exhibits the aim, the technology used and the environment under which the authors propose their solutions. Despite that the different proposals come from the related works, some questions remain open such as: How to integrate WS in dynamic environments in an autonomic way without losing the order of the messages otherwise keeping information congruent?

Many other approaches have and still are proposed for message ordering [11] [10]. However, in [11] to achieve such the author makes use of other software components
TABLE I: Related Works.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>1</th>
<th>13</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aim</td>
<td>Avoid erroneous collaborative scenarios</td>
<td>Integration of multiple collaborative systems, conference (control framework)</td>
<td>Integration of collaborative work environments</td>
</tr>
<tr>
<td>Technology used</td>
<td>Causality techniques and IDR</td>
<td>Web Services</td>
<td>REST Web Services</td>
</tr>
<tr>
<td>Environment</td>
<td>Distributed and heterogeneous engineering</td>
<td>Distributed and heterogeneous video conferences</td>
<td>Distributed and heterogeneous tools</td>
</tr>
</tbody>
</table>

like a message broker that temporally stores messages and processes them in the order in which they were received. In [11] the author presents a way of message ordering by using labels and proxies; however in communication both ends must agree on the initial label and sequence, then the proxy reads this label and re-orders messages if they arrive out-of-order. The industry has also attacked the problem of message ordering, for example Oracle@ [10]. Which proposes a strict message ordering using WebLogic JMS; this is a value added proprietary software. To ensure message order with respect to the processing order for a group of messages. The messages are stored grouping them in a single unit called Unit-of-Order.

IV. MESSAGE ORDERING FRAMEWORK (MOF)

Mission critical collaborative environments need a more reliable transport mechanism particularly where the order of the message matters, but JMS alone cannot grant such. The Message Ordering Framework (MOF) is built over JMS, extending its properties. Furthermore, all communications passing through it are enriched with small overhead, consisting on control information, keeping track of the order of messages. MOF uses the ESB messaging middleware over distributed heterogeneous networks to support the publish/subscribe communication model linking autonomously various publishers and subscribers. Additionally the ESB ensures interoperability and offers several features such as: service discovery, intelligent routing, message processing and service orchestration [6]. Also, it ensures the proper format between service providers and consumers, no matter which programming language they are written on [2]. MOF does not depend on the WS in place; to achieve such we present the scheme shown in Fig. 2 where the ESB system is in charge of the memberships of the users, subscribing and departing them automatically while maintaining causality properties as depicted in Fig. 3a for subscription and 3b for departure.

A. Membership subscription

Fig. 3a shows the subscription of a new user to the collaborative work environment. When a participant or user wants to join he must send an admission request to the ESB which lets other users know that a new user is joining with the $join_{request}(p_k,p_i,VT(p_i[\bar{\delta}]))$, where he is assigned its vector clock value. Finally, the user must send a $join(p_k)$ to the ESB, which sends it to all other users. Once received, the user is properly integrated to the collaborative environment.

B. Membership departure

Fig. 3b shows the departure of a user from the collaborative work environment. When a user wants to leave, first he must send a petition request $leave_{request}(p_k)$, which is sent by the ESB to all other users announcing that a user will leave the environment. Finally, the user leaving, sends the $leave(p_k)$ message to the ESB; once received by all other participants the user has a proper departure from the collaborative environment and thus causality is preserved among the rest of the participants.

C. MOF’s Architecture

The aim of the MOF is to propose an extensible framework for the JMS API which can be exploited by WS, particularly in collaborative work environments over heterogeneous networks; this is based on the IDR having low overhead and maintaining the informations’ coherence. A MOF is optimal because it transmits the minimal and necessary amount of control information to completely preserve the causal order among messages or events; also is able to manage interoperability and scalability because it is built for services and the IDR is designed to deal with large distributed systems.

Fig. 4a illustrates in detail the communications that take place in a collaborative work environment, where publishing
reserve memory space
MOF's Architecture
Web Services Call
JMS Adaptor
Causal Properties
JMS Listener
Publish /Subscribe

Fig. 4: MOF’s Architecture.

Fig. 3: Membership.

(web services) of a topic/queue takes place and subscribers (consumers of the service) can retrieve collaborative information. In either way publish or subscribe the communication passes through the ESB and then it is delivered to the MOF:

- The **WS Call** represents that WS are usually deployed in remote servers hosted by a third party. The ESB provides the WS callback mechanism as a proxy that can be accessed using JMS.
- The **JMS adapter** is needed to communicate with said proxy allowing the exposure of the WS as a request that is then passed to the **Causal Properties** component.
- The **Causal Properties** component is in charge of maintaining the order of messages. To achieve such messages requests or replies are enriched with the IDR. To keep it optimal, attaining minimality, timestamped causal information per message corresponds to messages linked by an IDR.
- The **JMS Listener** then listens to events and delivers or queues messages, that is, subscribes or publishes the message to a queue or topic.

V. Conclusions

In this paper we unveiled the Message Ordering Framework (MOF) for Web Services (WS) in distributed collaborative work environments based on the Immediate Dependency Relationship (IDR). It is of vital importance not to overwhelm the network, with unnecessary information and to properly use dedicated resources.

MOF can be used to keep the order of messages congruent in any collaborative environment that are WS-based. Our approach leverages the advantages of JMS keeping its main functionalities but at the same time making it more robust.

Acknowledgment

Special thanks to the Consejo Nacional de Ciencia y Tecnologia (CONACyT) for the grant granted for the fulfillment of this project.

References


