Building a Generic Real Time Collaboration Framework

Bruno Silva, Ricardo Preto
Ubiwhere

Abstract—**Current document contains the proposal of a solution that will allow building a generic real time collaboration framework. It is proposed that this framework uses a centralized architecture because it will allow using a central instance to solve all operation avoiding too many heavy and time consuming negotiation among multiple peers. A Central Server and a Client Library are proposed to be developed and will form the proposed framework. Central Server will be responsible to receive, resolve, persist and broadcast all operations made by users sharing a given resource. The modules that will form central server are specified attributing to each one the main responsibilities to be addressed. The structure of the document that will represent the shared resource is also specified in order to describe how operations will be treated, resolved and persisted. Operational Transformation (OT) will be used to apply all changes to the document. The version of the OT to be used will be Google Wave OT mainly because it is open source and was subjected to a heavy use having been the core of Google Wave. The Client library that completes the framework is to be used by third party applications in order to interact with Central Server. Third party applications will import this library and use its features to add collaboration featured to their implementations. Client library will be responsible to handle the communication and possible local and remote conflict resolutions sent and received from the server. The proposed framework was subject of an implementation using Java EE6 and GWT to build Central Server and Client library. Besides building the framework, a third party web solution was built in order to check if the use of the framework would indeed seamlessly allow adding collaboration features. The use of the framework was a success having proved that the proposed solution indeed works in a real case scenario. Possible future implementations are also presented in order to add even more value to the proposed framework.**

I. Introduction

With the constant and rapid growth of information systems both in terms of technology and infrastructures, virtual connections among worldwide users can now be established, providing rich and high quality virtual environments. Users demand for these real time collaborative environments is rapidly growing in order to be able to work collaboratively and produce content or exchange ideas without physical restrictions. However, the implementation of these features is both complex and time consuming, which implies that the transformation of a product that is designed for single use into a product that allows multiple users to work on the same resource at the same time using an internet connection is, at most cases, too expensive and thus unaffordable. The framework proposed at this document aims to turn the transformation of a single user product into a real time multi user collaborative product not only affordable but also simple.

Implementing and providing a framework able to support multiple applications with different demands and business logics brings challenges that must be taken into account at a design level. One of the first and main concerns is the fear of building a framework so generic that does not respond to the needs of any application without some tweaks or modifications. The proposed solution will take all the different design and implementation problems into account, specifying what concerns should be considered and presenting a strategy to solve them.

II. Proposed Solution

The current section will explain possible problems and solutions regarding the implementation of a generic framework that can be used to enable collaboration features to a wide spectrum of applications. It is expected that this framework can be used and re-used by multiple applications discarding constrains associated with each application's business logic. The Framework will be responsible for managing the connection between users interacting at a given resource, resolving, broadcasting and persisting all the operations made by each user. Third party applications that use the framework will only have to handle the operations made by local and remote users, updating the correspondent interface with the action took by either remote or local users. Third party applications will not have to handle possible conflicts resulting on the multiple edition of the same resource at the same time. Only persisted and conflict free operations will be passed to the third party applications.

By allowing applications to only focus on the logic associated with the transformation of an operation to an interface update, it is avoided the implementation of complex connection and conflict resolution management systems.

The current document does not contain the implementation of this framework. It contains however a detailed description of a proposal to build this framework, justifying each one of the choices being proposed both at a design and implementation level.

A. *Peer To Peer or Centralized Architecture*

Before starting to design the envisaged solution, a study was conducted in order to understand which one of the alternatives would best suit the framework. Both of them have strengths and weaknesses and the choice of which one to embrace will determine indefinitely and deeply the framework's behavior.

A P2P approach would allow the framework to not depend on the availability of a central service. The need for a central service to resolve and persist all users' operations involves the maintenance of an infrastructure that can handle multiple requests at the same time, keeping response time low. These costs must be taken into consideration because it is expected that the central service will handle multiple operations from multiple users in a multiple applications' context. Using a P2P approach would discard the need of a central service, thus reducing considerably the costs associated with the solution. It would also allow peers to quickly connect and collaborate without having to rely on the response time and availability of a centralized service. However, as it was previously stated, the use of P2P has also some constraints that must be taken into consideration. By using a P2P approach, clients would have to resolve all conflicts among peers, which entails some constrains that must be deeply analyzed. This scenario may not pose a constraint and even work great better than with a central service implementation for a low number of users. Nevertheless, if dozens of users start to collaborate on the same resource at the same time, all operations would have to be received, transformed and re-sended among all peers. If conflicts occur the cost of solving them would start to be notable by the final user since performance would greatly decrease. This is due to the fact that, without a central service resolving conflicts and having the final word on this resolution, negotiation would have to be made among peers in order to all of them agree on the final state of the resource after all operations are applied. This results on further communications among peers and further checks and processing on each one of the alternatives proposed by each peer.

The implementation of a solution designed with a centralized architecture allows the existence of a central service that is responsible to solve all the conflicts. This implementation avoids having multiple peers resolving the same conflicts and possibly generating further conflicts to be resolved. By having a unique entity solving conflicts, the final word/version is always the version persisted at the central point. Having a central point also allows persisting all information, enabling clients to disconnect and re-connect at a future point resuming all the work made while they were offline. If more than one client is sharing a given resource, if a client disconnects and posteriorly reconnects he/she will receive all the changes made by the other users while he/she is offline.

After considering the pros and cons previously stated, the following point were though to be essential:

- Using a central service to solve all operation conflicts
- Recover and receive all other clients' editions on the shared resource on reconnect without depending on the availability of other peers

Both points are associated with a centralized architecture and so this approach will be used to design the proposed solution.

B. *General View*

Just as previously stated, the proposed framework will use a Centralized Architecture approach. The following image displayed a general view of the proposed solution.

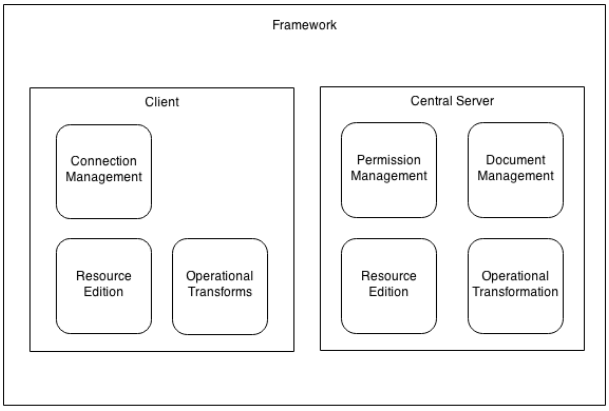

Figure 1 - General View

As can be seen in the previous figure it is proposed that the Client and Central Server modules compose the framework. Central Server will be a standalone service allowing multiple applications to interact with it using the Client library. The connection will be established using sockets that will provide a bidirectional channel between Central Server and Client.

1) *Central Server*

The first and main module that will be needed in order to build the proposed solution will be a central server to resolve all concurrency. This will be an independent standalone service that will allow multiple clients from multiple applications to interact with it, solving and storing all received operations, broadcasting and returning the result to all connected client's sharing a given resource. In order to connect with central service, clients will have to establish a valid socket connection with the central server in order to open a bi-directional communication channel. Central Server's main features will be:

- Provide a set of features that allow to concurrently edit a resource by multiple users
- Persist all operations made by all users to a given resource
- Solve all conflicts that result by the application of multiple operations to the same resource at the same time
- Control the access to the resources

- Provide a set of document management related features

All these features must allow the implementation of a real time collaboration framework to different applications without having to make any adjustments. Previous features are presented at Figure1 being contemplated in the sub-modules present at Central Server.

The first thing that must be taken into consideration is how a given resource will be identified. It is proposed that the responsibility of the definition of this id is held by the central server in order generate a unique identifier each time an application creates a new shared resource. A set of features associated with document management will have to be provided by the central server enabling third party application to not have to implement any of these features. These features will be agglomerated by Document Management sub-module and will contain:

- Create shared resource
- Delete shared resource
- Edit title and description of the resource

Permission management will always have to be made by the central server in order to verify every time a new share request is received if the application/user has the permission to see and edit the targeted resource. The following set of features are proposed for the sub-module Permission Management:

- Register new user
- Recover password
- Login
- Logout
- Add user to resource (Allows target user to see and edit shared resource)
- Remove user from resource

As previously stated, Central Server will have to handle all operations made by users to the resource. In order to do so, it is required a specification of the generic resource structure and supported operations. The proposed structure resembles the HTML structure of a web page containing elements that can include other elements or text elements. At the top of the resource persistence document, the parent element will be placed and this element will contain as attributes documentId and resourceDescription. These attributes will not be concurrently edited, being only defined through document management features. This element will have the tag name “ResourceElement” and just like an html element will have a start and end tag. Bellows there is an example of the final result of the creation of an empty document:

```
<ResourceElement documentId=”uniqueId1”
resourceDescription=”ElementDeTeste”/>
```

This element will contain all other elements to be added to the resource. In order to not restrict the use of the framework to any type of application the following elements are proposed to be possible to be added as children of the parent element:

- ResourceElement

  Allows adding an element that has attributes that resembles attributes of the parent top element. This element will be easily identified at the document by providing the documentId. Further attributes can however be added to this element. Furthermore, it will be possible to return the content of this “sub-resource” allowing creating collaborative resources within collaborative resources. This element will also allow other elements to be added as children containing no restrictions when comparing to the resource parent element. The tag of this element will be “ResourceElement”.

- ContentElement

  This element will allow adding any attribute not containing any attribute by default. It will also allow other elements to be added as children. The tag of this element is not restricted and so can have any value that will be defined on element creation.

- TextElement

  This element will allow adding any attribute not containing any attribute by default. The content of the element will be a text value. Operations of text append and text remove will be allowed in order to edit the content of this element. The tag of this element is not restricted and so can have any value that will be defined on element creation.

An example of a resource document that can be used by a real time text edition application is presented bellow:

```
<Resource documentId=”uniqueId1”
resourceDescription=”ElementDeTeste”>
<user1 caretPosition=”10”/>
   <user2 caretPositon=15/>
   <user3 caretPositon=11/>
   <Paragraph fontSize=”21” fontStyle=”Arial” fontColor=”Blue”
fontWeight=”Bold”>
      Title of the document being currently edited
   </Paragraph>
   <Paragraph fontSize=”14” fontStyle=”Arial” fontColor=”Black”
fontWeight=”Bold”>
         Text of the first paragraph being concurrently edited
   <Paragraph fontSize=”14” fontStyle=”Arial” fontColor=”Red”
fontWeight=”Bold”>
      <line>A second paragrapgh with a different style</line><line
fontColor=”green” >that even changes style in the middle of the
line</line>
   </Paragraph>
</Resource>
```

Another example of the resource document of an application that allows to two players play chess online:

```
<Resource documentId=”uniqueId1” resourceDescription=”chess
game”>
   <Player1ActivePieces>
      <King positionX=”5” positionY=”1”/>
      <Pawn1 positionX=”5” positionY=”3”/>
      <Pawn2 positionX=”8” positionY=”3”/>
   </Player1ActivePieces>
   <Player2ActivePieces>
      <King positionX=”2” positionY=”7”/>
      <Knight1 positionX=”1” positionY=”8”/>
      <Bishop2 positionX=”4” position=”7”/>
   </Player2ActivePieces>
   <Player1LostPieces>
      <Pawn3/>
      <Pawn4/>
      <Pawn5/>
      <Pawn6/>
      <Pawn7/>
      <Pawn8/>
      <Knight1/>
      <Knight2/>
      <Queen/>
      <Rook1/>
      <Rook2/>
      <Bishop1/>
      <Bishop2/>
   </Player1LostPieces>
   <Player2LostPieces>
      <Pawn1/>
      <Pawn2/>
      <Pawn3/>
      <Pawn4/>
      <Pawn5/>
      <Pawn6/>
      <Pawn7/>
      <Pawn8/>
      <Knight2/>
      <Queen/>
      <Rook1/>
      <Rook2/>
      <Bishop1/>
   </Player2LostPieces>
</Resource>
```

As it can be seen by the two previous examples, the framework supports multiple applications with different business logics and purposes. More examples could be presented but the generic features are well displayed from the previous two examples. In order to concurrently manipulate a document with the previously presented structure, the following features are proposed:

- Add ResourceElement
- Edit ResourceElement description
- Remove ResourceElement
- Return ResourceElement state
- Add ContentElement
- Remove ContentElement
- Add TextElement
- Remove TextElement
- Add text to TextElement
- Remove text from TextElement
- Add attribute to element
- Remove attribute from element
- Edit attribute of element

The implementation of all these features will have to take into consideration possible conflicts that result from the manipulation of the same objects by multiple users. Possible troubles and proposed implementation will be detailed bellow in **Concurrency Control** subsection. All possible actions of third party application will have to be translated using these features that will be provided by the Client library still to be presented.

2) Client

It is referred previously that the proposed framework will allow to effortless add collaboration features to any application without the need of further adjustments. Besides this, third party applications are supposed to not have to handle the communication with central server neither resolve concurrency conflicts that are generated from local and remote operations being applied to the same resource/element at the same time.

To make sure the previous statement is true, client will have to be responsible for handling the communication with the central server and translating user actions to the pre-defined resource actions supported by the server. These concerns are to be addressed by the modules Connection Management and Resource Edition displayed at the Figure 1.

Connection Management is responsible to connect, disconnect and manage the connection with central server. It will provide features that will enable third party applications to login, logout, register new users and recover user passwords. Besides this, it will implement an important feature that will allow user to produce work while offline. Despite the fact that nowadays Internet connections tend to be quick and stable, it is also inevitable that in some cases users experience Internet connection failures. This module will provide an important feature that will enable users to keep editing the shared resource while offline, being those changes send to the central server when the connection is reestablished. The Client library will notify the third party application that changes are not being stored at the central server but will allow to continue receiving operation and by doing so will not block user’s operations. When connection is reestablished this module will send all pending operations, being those operations resolved and persisted by the central server. Central Server will then return the operations’ result as well as all pending operations from other users made while the user was offline.

Resource Edition module will be responsible for providing all the features regarding resource concurrent edition available at the central server. It will be responsible to receive users´ operations and construct the correspondent valid operation to send it to the Connection Management module that will send it

to the server. It is intended that the third party application does not have to implement any operation conflict resolution feature, being only notified of other user's operations and conflict resolution operations applied to local operations. In order to fully implement these features and provide a truly black box service, Client module cannot be a simple proxy that sends and receives operations to and from the central server. If this approach were to be implemented, in case the connection with central server fails, third party user application interface would have to be blocked waiting for connection to be re-established. Besides this, if connection was slow or central server response took a perceptively time to respond, user would experience a considerable delay when waiting for local operations to be sent, resolved, persisted and returned. Because of this it is proposed that the client also has a local resource state that is updated using the same conflict resolution algorithm present at the central server. Both local and remote operations will be applied to the local resource document. Client will have a version of the shared resource on which will apply local and remote operations prior to send local changes or locally notify remote changes. This implementation will allow saving time of central server solving conflicts that can be locally solved. If multiple remote operation are received from central server and at the same time a local operations is applied, this module will apply the same conflict resolution algorithm available at the central server to infer the final state resulting from the application of both remote and local operations. Only one operation will be sent to the central server that will hopefully (the normal scenario) be a conflict free operation that will take into consideration both remote and local operations. By doing this, it avoids central server to have to solve a conflict needing only to persist a conflict free operation. When considering scenarios with many users, the fact that local clients can send conflict free operations is a huge benefit for the central server. It is however to note that Central Server version will always be final and so, if a conflict emerges on the Client's version when comparing with the Central Serve version, it will be the Client's version that will have to adapted in order to reach Central Server's version.

It is intended that the Client is to be provided as an external library to be added to any third party application. Third party application will use all public features provided by this lib to interact with central server. In order for the Client library to interact and notify the third party application, an interface implementation will have to be provided by the third party application implementing all notification methods that will be used by the Client library. Despite the fact that it is intended that the final framework is generic allowing that any application can use its features, the proposed implementation poses a restriction. Both third party application and the developed Client library need to be compatible. Despite the fact that this may restrict the use of the client in some applications, it is also true that if the client library is incompatible with a targeted application's technology, a new version of the client library can be developed without having to change the central server service.

As it was previously stated, client will implement the same behavior of the central server regarding operation conflict solving. Implementation of these feature is analyzed at the subsections **Concurrency Control** available bellow.

C. Concurrency Control

Concurrency control will allow both client and server to solve any conflict regarding the sharing of a given resource by multiple users. In order to implement all the real time collaboration features Operational Transformation (OT) implementation will be used. The main reason to use OT as the concurrency control framework is the widely research already made and available to all (this theoretical framework is being actively researched for more than 10 years) and the fact that is widely used by many applications (SubEthaEdit, EtherPad, Google Docs, Mockinbird, CoPowerPoint, CoMaya, Apache Wave) that already provide a real time virtual environment to multiple users. The widely use of OTs among these solutions proves not only that OTs algorithms are reliable but also that they can handle a heavy load of operations made by multiple users editing the same resource.

In order to use OT, all central server and client resource edition features will have to transform each action to an OT operation that will then be applied to the shared resource. Just as previously said, there are innumerous OT implementations already available that can be used. In order to build the framework being targeted at the current document, Google Wave OT implementation is proposed. The main reason for choosing this implementation is the fact that Google Wave OT was the core of Google Wave having been subjected to a considerable higher heavy load than other available implementations. Furthermore, the implementation is not only open source and available to all but is also being maintained by the Apache Foundation under the project Apache Wave. Apache Wave Foundation has currently an active community working on Apache Wave that is maintaining Apache Wave and is open to help in any doubt regarding the use of Apache Wave and the implementation of Google Wave OT algorithm. If these reasons were not enough, Google Wave OT also provides important features that can be seamlessly added to central server. Listed bellow are the features though to be relevant to the proposing framework and that are also in favor of choosing Google Wave OT:

- Undo/Redo features. Google Wave OT implementation already has this feature embed
- Composition. Google Wave OT implementation allows taking two or more operations and merging it together into one single operation. This will allow client and server implementations to avoid sending multiple operations when only one operation can be sent to all connected clients or server.

The implementation of Google Wave OT will have to follow the same methodology both in client and server. The proposed resource structure is completely compatible with Google Wave OT algorithm and so no adaptions will need to be made. Current document will not describe neither the basic

theory of OT neither the changes proposed by Google Wave OT because this information is available at multiple sources that deeply specify and document each approach.

D. Final Considerations

With all previous information, and in order to start implementing the framework, only technologies need to be defined to start working. This however will not be specified because by defining both central server and client technologies we are adding a severe restriction to the framework implementation and future use. The selected technologies will have to take into consideration the know-how of the developing team, the targeted usage scenarios, the targeted infrastructures to place the central server, among other possible reasons. Besides this, all previous specification does not account any restrictions regarding technology choices and so this restriction will not be placed by risking proposing a given language/technology.

E. Building the Solution

After specifying the proposed framework, and in order to analyze if what was being proposed really delivered what was intended, a framework and a third-party application were developed.

1) Framework

The developed framework used all what was previously defined in its implementation. By following all what was previously specified a server instance was developed using Java EE 6 that allows clients to connect using a socket connection. It receives, resolves and persists all operations made to a given resource by authenticated users. Google Wave OT algorithm was added to the server and it is used to apply and resolve all received operations.

Client library of the framework was developed using Google Web Toolkit. GWT is an open source framework that allows creating and maintaining a Javascript web application using Java. By using GWT it was possible to seamlessly port all Google Wave OT algorithm from central server to the Client library.

These choices should not be taken as restrictions to implement the proposed framework. These technologies were only used because development team has high experience in using both Java and GWT.

2) Third Party Application

In order to use the developed framework a third party application was developed on which collaboration features were to be added. Just like previously said, it was expected that the developed framework was able to add any to application real time collaboration features. Many applications could be developed but development team felt that a real time collaborative whiteboard would be the perfect test to the framework. This choice was mainly due to the heavy load of operations that the framework would have to handle when multiple users started drawing lines on the whiteboard.

The final product is a web solution that uses GWT and provides a whiteboard allowing users to edit it's content in real time at the same time. Users can create new documents and invite other users to a given document allowing them to edit it's content. In order to provide a collaborative whiteboard a HTML5 canvas is used allowing users do draw using mouse or touch events. An adaptive design was implemented being the solution prepared to be used by smartphones, tablets, desktops and smart TVs adapting it's layout and input events to the target device. Features of drawing a free line, remove a line and change color were added to the solution in order to interact with the canvas.

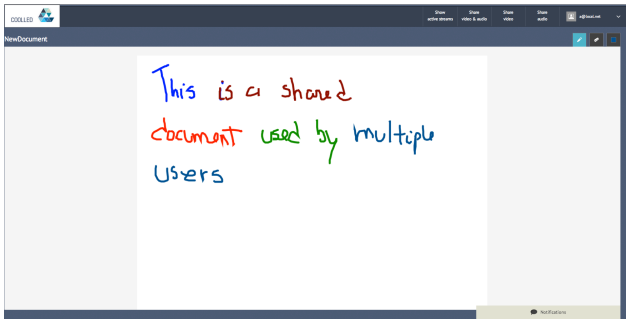

Figure 2 - Coolled whiteboard page

Before using the developed framework, the solution provided a standalone whiteboard on which a user could draw editing its content. After reaching this step, the solution became a perfect real case scenario to use the collaboration framework. It was expected that by using the client, only implementation regarding the update of the canvas with local and remote operations would have to be handled. By using the framework, this was indeed what happened. Client library was added to the project allowing easily creating a connection with the server. Client's Document management features are used to create and share documents among users. All local operations on the canvas that are retrieved by touch and mouse events, are passed to the Client library that is responsible to convert into operations and persist them in the local copy of the shared document. After being locally applied (without having confirmation from the server at this point) Client library notifies third party application of the changes to be added to the whiteboard (Canvas). Client is then the only responsible to send these operations to the server and manage possible conflicts. All messages received from the server (being local operations resolution/acknowledge or remote operations) are processed also by the Client library. Third party application is only notified when an operation is correctly locally persisted notifying if an element was created/edited/removed. Third party application only has to update user interface taking into consideration the information received.

In order to implement the real time collaboration features, only one type of element (TextElement) is being currently used. Bellow is the example of a document that represents a shared resource containing two lines each one containing three points:

```
<Resource documentId=”w+efe1b6b524c8b4b41e060f79d6e645f5A”
resourceDescription=”WhiteboardTest”>
   <Stroke color=”#FF9966”>
      1,1;2,2,2,1;2,2
   </Stroke>
   <Stroke color=”#FF1136”>
      122,122;122,122;123,122;123,124
   </Stroke>
</Resource>
```

As can be seen, TextElements are used to represent each one of the lines. A line is represented by an element with a tag name “Stroke”. The content of the element will contain the points that compose the line separated by “,” and separated by “;” from each other. When users are creating a line, points are appended to the correspondent Stroke element using an OT transformation that appends the new points to the end of the actual content of the element. The attribute color of each Stroke element will be used to persist the color of the stroke. In order to locally create a line, third party application will only have to use Client library function that allows creating a TextElement defining the tag name as “Stroke” and adding the current color as an attribute of the element. After this creation, it will use the function that allows adding content to a TextElement passing the points as text separated by “,” and “;”. In order to update whiteboard with remote operations, third party application will only have to be notified by Client library that either an element creation or edition with a tag name stroke was received. If this is true it will process the correspondent whiteboard update taking into consideration if a new line is to be created or points should be appended to an existing line. It uses the value of the attribute color to know which color will be used to represent the line and the content of the stroke element to know the line’s points to be added to the canvas. Third party application does not have to handle any conflict resolution or connection management events. Client implements all these features informing only the third party application when a new event was persisted/received.

F. Future Work

The main objective of the proposed framework was to allow real time collaboration features to any application regardless application’s business logic. Although specification and posterior implementation were a success, there is still work to be done in order to embed important features to the framework adding even more value to the final solution. The proposed framework has the potential to be provided as a SaaS. In order to do so, some changes would have to be made to allow controlling and distributing central server resources among the different clients. The first changes would have to be in the document creation and system management modules. Client registration and user management should be associated with a given application and by so a module allowing applications to be registered and managed would have to be created. Furthermore, the traffic associated with each application would have to be monitored in order to be charged. A module responsible to do this would also have to be specified and implemented.

The developed whiteboard solution has also the potential to be worked in order to provide the full set of features associated with the scenario of multiple users editing a canvas. Shapes and textboxes for instance could easily be added being both changes supported by the framework though the use of ContentElement and ResourceElement.

G. Conclusions

The proposed framework specification allowed building a generic framework that allows implementing collaboration features to any application without modification to the framework. The degree of specification present at the document proved to be a valid approach on building the proposed framework without needing any further specification and changes on building the final solution.

The integration of the framework in a real case scenario was a success being the framework able to handle all the targeted features. The main work developed at the whiteboard solution was centered on the implementation of the solution’s desired features without having to implement any feature regarding conflict or communication management among peers editing the same resource. It is therefore safe to say that the framework did meet all the initial expectations and if in a near future real time collaboration features are to be added to a given application, the use of the specified and developed framework will be almost certain.

REFERENCES

[1] C. Cook, N. Churcher, “A user evaluation of synchronous collaborative software engineering tools”, 2005

[2] Jae young Bang, Daniel Popescu, George Edwards, Nenad Medvidovic, Naveen Kulkarni, Girish M. Rama, Srinivas Padmanabhuni, “A Highly Extensible Collaborative Software Modeling Framework”, 2010

[3] M. Cataldo, Al. Camel, “A tool for collaborative distributed software design”, 2009

[4] David Wang, Alex Mah, Soren Lassen, “Google Wave Operation Transformation”, July 2010

[5] R. Hedge, P. Dewan, “Connecting programming environments to support ad-hoc collaboration”, 2008

[6] Carl Cook, Neville Churcher, “Modelling and Measuring Collaborative Software Engineering”, 2005

[7] Michael Reeves, Jihan Zhu, “Moomba A Collaborative Environment for Supporting Distributed Extreme Programming in Global Software Development”, 2004

[8] C. Cook, N. Churcher, “Towards Synchronous Collaborative Software Engineering”, 2004

[9] N. Graham, H. Stewart, A. Ryman, R. Kopaee, R. Rasouli, “A World-Wide-Wb Architecture for Collaborative Design,” 1999

[10] J. Hill, C. Gutwin, “Awareness Support in a Groupware Widget Toolkit”, 2003

[11] David A. Nichols, Pavel Curtis, Michael Dixon, John Lamping, “high-Latency, Low-Bandwith Windowing in the Jupiter Collaboration System”

[12] James Begole, Mary Beth Rosson, Clifford A. Shaffer, “Flexible collaboration transparency: supporting worker independence in replicated application-sharing systems”, 1999

[13] C. A. Ellis, S. J. Gibbs, “Concurrency control in groupware systems”, 1989

[14] Du Li, Rui Li, “Transparent sharing and interoperation of heterogeneous single-user applications”, 2002
[15] Chengzheng Sun, Clarence Ellis, “Operational Transformation in Real-Time Group Editors: Issues, Algorithms, and Achievements”
[16] Gérald Oster, Pascal Urso, Pascal Molli, Abdessamad Imine, “Real time group editors without Operational transformation”, 2005
[17] Clarence Leung, “Operational transformation in cooperative software systems”, 2013
[18] Marcos Bento, Nuno Preguica, “Operational transformation based reconciliation in the FEW File System”, 2006