International Journal of Networked and Distributed Computing, Vol. 2, No. 2 (April 2014), 91-99

On a Drawing-Frequency based Layered Canvas Mechanism for Collaborative Paper Editing Support Systems

Shinya Katayama [1] , Takushi Goda [2] , Shun Shiramatsu [1] , Tadachika Ozono [1] , Toramatsu Shintani [1]

[1] Dept. of Computer Science and Engineering, Graduate School of Engineering,
Nagoya Institute of Technology,
Nagoya, Aichi, Japan
E-mail: katashin@toralab.org, {siramatu, ozono, tora}@nitech.ac.jp
[2] Dept. of Computer Science, Nagoya Institute of Technology
Nagoya, Aichi, Japan
E-mail: godata@toralab.org

Abstract

We propose a Drawing-Frequency based Layered (DFL) Canvas approach to reduce a synchronization delay in collaborative web applications. The synchronization delay consists of a network delay and a drawing delay. The network delay is well-argued topic by various researches. We focus on the drawing delay, which is a primary bottleneck of Canvas objects synchronizations. We introduce the DFL Canvas approach in order to solve the drawing problem. The DFL Canvas divides Canvas objects by drawing-frequency. We implemented a collaborative paper editing support system with the DFL Canvas. Users can edit the same document and share Canvas objects on papers in real-time on web browsers. We present an evaluation result that indicates the approach is suited for collaborative paper editing support systems.

Keywords: Drawing-Frequency, Layered Canvas, Collaborative Web Application, Synchronization

1. Introduction

We have developed a collaborative paper editing support system.[1] On the system, users edit papers and synchronize Canvas objects, defined in HTML5. We propose a Drawing-Frequency based Layered (DFL) Canvas approach for collaborative web applications. The DFL Canvas reduces a delay of Canvas synchronizations and we apply the DFL Canvas to the collaborative paper editing support system.

A paperless environment is recently an attempt to distribute, manage, or read digital documents. Recently, documents or papers are made as digital documents. More and more digital documents are distributed in conferences. Companies can decrease the cost and optimize the work by promoting paperless. The maintenance can become much easier with the use of efficient search and structure in paper-based filing system.

A paperless system is applied to the educational field.[2,3] Students send paperless reports to a professor. A professor comments on and scores paperless reports. Students discuss a study or an experiment in a text chat. Professors and students reduce their bur-

Copyright: the authors

dens of manually writing and commenting on a report. Moreover, Professors can lecture remotely.

There are various researches for collaborative web applications. In the collaborative web applications, everyone use the web applications with no setup cost. Moreover, the web applications serve appropriate user interfaces for different devices. Collaborative web applications for editing require a synchronization of objects, e.g. DOM elements, which have texts, images or so on. The length of delay on a DOM element synchronization depends on the size of a synchronized DOM element and a network delay. The problems are well-argued by various researches that have proposed Operational Transformation (OT) based approaches.[4,5,6] However, the OT based approaches cannot solve the Canvas drawing delay problem, because a Canvas DOM element should clear and redraw all of the drawings on updating or removing its some drawings. Hence, We need to develop a novel Canvas synchronization approach. In this paper, a Canvas and Canvas objects (or drawings) mean a Canvas DOM Element and a drawing on a Canvas DOM element respectively. In this paper, a Canvas contains Canvas objects.*

Some researches propose a transformation approach to make collaborative web applications.[7] The transformation approaches for web applications turn single-user editors into multi-user editors supporting shared editing. We have also developed a real-time collaborative web page editing system based on transformation approach, WFE-S, which focuses on DOM-based collaborative editing.[8] WFE-S is a system to edit collaboratively existing web pages, and uses a web browser bookmarklet for feasible web page editing. We proposed a DOM elements synchronization mechanism on collaborative editing with WFE-S.[9] The existing approaches do not focus on the problem and our approach described in this paper can be applied to them to solve the problem.

In collaborative work, users share documents, applications, and so on. Kawashima and Ma[10] built a collaborative platform, TOMSCOP. Users can share web documents by using a shared web browser in the system. The user can communicate with other users by text chatting with a shared application. We aim to share and edit documents of meetings, conferences, or any other collaborative works. Users can add annotations on documents with our web application for collaborative works. Our approach is also applicable for a web annotation system[11] easily.

2. Related Works

We have developed the Wisdom Web Conference (WWC),[12] which is an iPad application for paperless meetings. A client of WWC gets documents from a server. A user can read documents through WWC. WWC shows a mark by touching and holding a display. We called the mark a “pointer”. The pointer helps communicating in a meeting by being synchronized with other clients. WWC has the speaker mode, the listener mode, the free mode, and the pointer mode. The client in without the free mode synchronizes the pointer. A transition of the page by the speaker mode client synchronizes with the listener mode client. Meeting participants feels more confortable with the synchronization of the page transition. In this paper, we propose a new application on the web. The user can use the web application with less cost because the application runs on an existing web browser.

There are some web applications or web platforms for collaborative work. Sols et al.[13] propose to use spatial hypertext layers as the solution for the management of annotations in wikis. Lo et al.[14] aim to build a web application in which users can get feedback on their English sentences. Users edit a text document on the web collaboratively, add or delete sentences and add annotations on web application. A web application tends to show low performance when editing a document. For example, a user cannot draw on any coordinate of a document. In this paper, we apply Canvas API to our web application for improving the performance.

A lock system is one of the important mechanisms for a synchronized application. Song et al.[15] system uses a file locking mechanism for collaborative work.

*In fact, a Canvas DOM element does not have Canvas objects as child DOM elements.

Copyright: the authors

They aim to implement a general collaborative editing platform, which supports collaborative editing of any type of documents by using users familiar software. The file locking mechanism is useful to manage consistency of an edited document. However, the user cannot edit the document that is edited by another user. We apply an annotation locking to the web application to solve the problem. It is necessary that a client can recognize timing of the beginning of an editing and the end of an editing. We propose a consistency management mechanism. The consistency management mechanism is applied to the annotation locking.

3. Drawing-Frequency based Layered Canvas (DFL)

We propose a Drawing-Frequency based Layered (DFL) Canvas approach for a fast synchronization. Users enable to synchronize Canvas objects with lower delay on the DFL Canvas. The DFL Canvas divides Canvas objects by a drawing-frequency. We divided the most frequently updated Canvas objects into the real-time layer. Moreover we also divided the not frequently updated Canvas objects into the mutable layer and the immutable layer, because of a number of the objects. Fig. 1 shows the overview of the three types of Canvas layers.

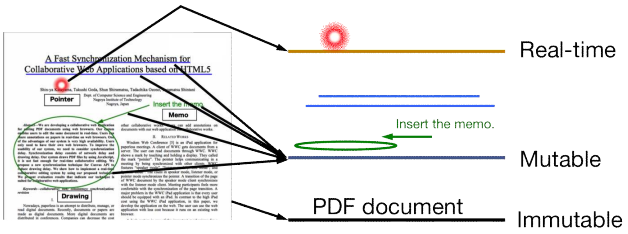

Fig. 1. The Three Types of Canvas Layers.

Users enable to synchronize Canvas objects with lower delay on the DFL Canvas. The DFL Canvas solves a synchronization problem of the Canvas API. The problem is the cost to redraw a Canvas, because a Canvas cannot redraw just one Canvas object. For example, when a user adds Canvas objects *A*, *B* and *C* to the same Canvas, the Canvas redraws all of them when the user edits *B*.

We used a drawing-frequency to divide Canvas objects, because of the cost to redraw a Canvas. The Canvas redraw is a bottleneck of synchronization on collaborative web applications. The collaborative web applications synchronize their Canvas objects and redraw all Canvas objects when the objects are edited. The Canvas objects include various drawing-frequency objects, and a higher drawing-frequency object enforces wasted redraws to a lower one. Therefore, we focused on reducing the number of redraws for lower drawing frequency objects. We divided the Canvas objects by drawing-frequency in order to reduce the number of the redraws.

We divided the frequently updated Canvas objects into the real-time layer to reduce a cost to redraw a Canvas. The cost to redraw a Canvas increases in proportion to the number of Canvas objects. To reduce the cost, we divided Canvas objects by drawing frequency. The frequently updated object enforces a Canvas to redraw frequently. Therefore we divided the frequently updated objects into the real-time layer.

We also divided the not frequently updated Canvas objects into two layers: the mutable layer and the immutable layer, because of a number of the objects. The mutable layer has editable objects that are not frequently updated. All of the objects in the mutable layer are redrawn with updating an object. Therefore the mutable layer should have fewer objects as possible. To reduce the number of objects in the mutable layer, we defined the immutable layer. The immutable layer has uneditable objects and prevents the objects from being redrawn. Because of the immutable layer, users can avoid wasted redraws with uneditable objects.

4. Collaborative Paper Editing Support System

We implemented a collaborative paper editing support system with the DFL Canvas. Users can edit the same document and share Canvas objects on papers in real-time on web browsers. In the system, we realized the immutable layer as a document viewer, and the mutable layer and the immutable layer as an annotation viewer on the DFL Canvas. The system can configure

Copyright: the authors

a synchronization mode in order to collaborate various situations such as collaborative editing, presentations, and so on. The system manages a consistency of annotations, which are Canvas objects to edit papers on the system. Fig. 2 shows the system architecture. Fig. 3 shows the user interface of the system.

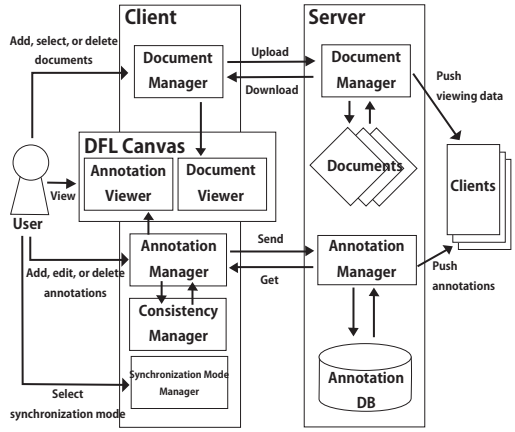

Fig. 2. System architecture.

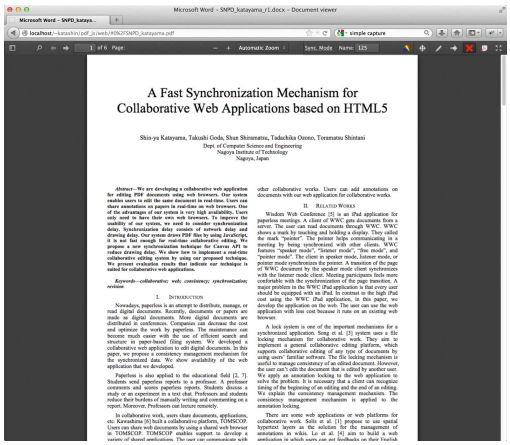

Fig. 3. User interface of the system.

4.1. Document Viewer & Manager

A document viewer, located at the center of the application, shows a document. Documents are stored on a server. A client downloads the documents when a user connects to the application. Furthermore, the client stores the documents to a local database. The local database is an HTML5 technology. There are three APIs for HTML5 local database, which are the Web Storage, the Web SQL Database, and the Indexed Database API. We use the Indexed Database API. Users select a document from a document selector. Fig. 4 shows the document selector. The document selector allows users to view all documents on the server. The document viewer shows the selected documents, which are fetched from the local database or the server. The document manager will downloads the selected document from the server and add the document to the local database if it is not stored yet.

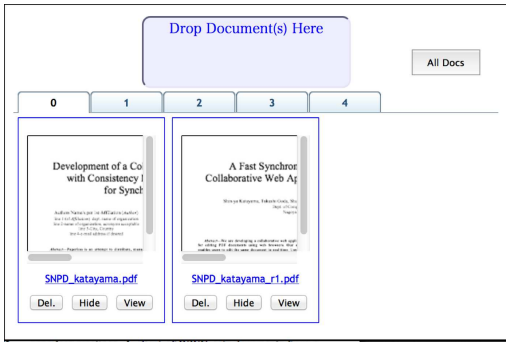

Fig. 4. Document selector.

The document viewer uses PDF.js to display PDF documents. PDF.js is a JavaScript PDF renderer with the technology of HTML5. The feature of PDF.js is that it can render PDF documents without native code assistance. Generally, users view PDF documents with plugins such as Adobe Reader. By using PDF.js, users do not have to install plugins to their web browser because PDF.js renders a PDF document with JavaScript and HTML5 Canvas API. Developers can apply additional HTML5 technology when rendering with PDF.js. We apply an annotation viewer on the document viewer for collaborative works.

4.2. Annotation Viewer & Manager

In the collaborative paper editing support system, users can use four types of annotation that are pointer, memo, drawing and stamp. An annotation on the collaborative paper editing support system is an object that is added on a document. Fig. 5 shows annotations on a document.

Copyright: the authors

The pointer is the circular, red object in Fig. 5. The user shows the pointer on a document by dragging. The pointer is synchronized and showed on all clients. The user can give a presentation or collaborate with the other users by using the pointer. The pointer is rendered on the real-time layer because the pointer requires a real-time render. The memo is used to present a text. The memo allows the user to discuss or revise a document. By selecting "Set Memo" on context menu, the user can set the memo on a document. By double-clicking the memo, the user edits the text. The drawing includes a line, an arrow, a circle or a rectangle. For example, the drawing can give attention to a sentence, and revise an incorrect word for document correction. Moreover, the arrow enables to connect the memo to a sentence. It is necessary because of space limitation. The user cannot always put the memo at a point where he/she wants to because there is limited white space on a document. The stamp is a figure or a drawing that is uniform shape. By selecting "Set Stamp" in the context menu, the user sets the stamp on a document.

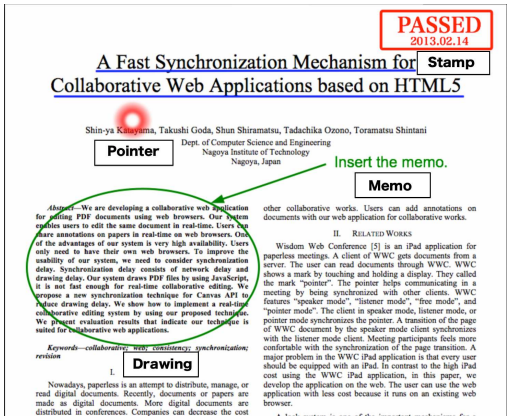

Fig. 5. Annotations on a document.

A user can set an annotation to any coordinate on an annotation viewer. The annotation viewer is on the document viewer. Users can substantially view annotations on documents. Users can customize the color of annotations by using an annotation color selector. The annotation color selector is popped up by clicking on the top-left button. A user selects an annotation color from color buttons or a color picker. The user can select various colors.

An annotation manager sends the annotation to a server, and the server stores it to a database. We use the SQLite3 database. A database table is used to create each type of annotation and each document. For example, the server creates a memo table named "memo_meeting.pdf" that is associated with the document named "meeting.pdf". A client sends an added, edited, or deleted annotation to the server. The server pushes the annotation to all clients. Therefore, the annotation is synchronized among all clients. A communication protocol for synchronization is WebSocket. WebSocket is a two-way synchronous communication protocol. The server enables to push data to the client with WebSocket.

4.3. Synchronization Mode Manager

A user switches the synchronization mode with a synchronization mode manager. Fig. 6 shows the synchronization mode manager.

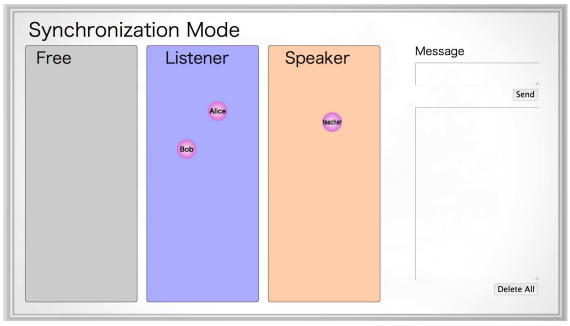

Fig. 6. Synchronization mode manager.

Let a circular object that has a text label be defined as a marker. The marker is related to the client. The marker represents a client identifier. A user sets the marker to identify the client on the synchronization mode. A rectangle that the markers ride on is a mode change area. The user sets the marker on different areas for different modes. Users switch the synchronization mode by dragging and dropping the marker on the mode change area.

There are three synchronization modes: the

Copyright: the authors

speaker mode, the listener mode, and the free mode. Each client can be set to any mode. Users change the type of synchronization mode depending on their purpose. For example, users switch to the speaker mode to give a presentation, or switch to the free mode to only read a document. A speaker mode client synchronizes positions of scroll, a magnification, and current viewing document to all listener mode clients. Free mode and the other two modes are independent and do not affect each other. However, annotations are synchronized between all clients even if they are in different synchronization mode.

Users can text chat with each other through the synchronization mode manager. The text message in the synchronization mode manager has a timestamp of when it was sent. The text message is synchronized in real-time among other clients. Users can communicate with each other by text chatting.

4.4. Consistency Manager

We use a lock mechanism to retain consistency in a synchronized data. The synchronized data is the annotation data that is synchronized among clients. An inconsistency may occur with users when editing the same annotation. In this inconsistency, an annotation's coordinate or text does not match among all clients.

A start of edit locks an annotation, and an end of edit unlocks the annotation. Therefore, it is necessary that a client recognizes the start and end times for editing respectively. Editing includes a move of coordinates and an update of a memo's text. Therefore, the client enables to recognize the time by catching mouse-up events or mouse-down events. A user updates the memo on a memo update form. The memo update form is displayed by double-clicking on a memo, and hidden by clicking on an update button or a cancel button. We clock the start time when a client displays the memo update form, then clock the end time when a client hides the form.

5. Evaluation

We evaluate the synchronization delay of Canvas objects. The synchronization delay is defined as the difference of drawing time between a drawing client and a synchronized client. The synchronization delay consists of a network delay and a drawing delay. This evaluation shows that the synchronization delay is rather small for collaborative web applications. We use a desktop computer for evaluation, equipped with Mac OS X 10.7.5, 2.7 GHz Intel Core i5 CPU and 4GB 1333 MHz DDR3 RAM.

5.1. Comparison of Single, Full and Drawing-Frequency based Layered Canvas Approach

We evaluated synchronization delays by redrawing Canvas of three approaches: a Single Layered (SL) Canvas approach, a Full Layered (FL) Canvas approach and the Drawing-Frequency based Layered (DFL) Canvas approach. In the SL Canvas approach, a Canvas just has one layer that contains all Canvas objects. In the FL Canvas approach, a layer just contains one object and is created when users add an object. The DFL Canvas is our approach, which proposed so far in this paper. For the synchronization delays, we calculate the delays of synchronizations when the client edits a Canvas object that is selected randomly from all Canvas objects. Fig. 7 shows the comparison of three approaches.

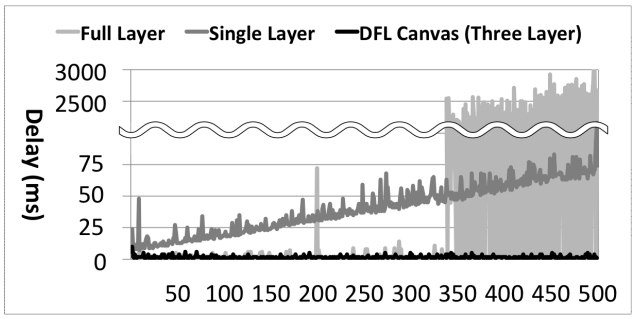

Fig. 7. Synchronization delay of SL, FL and DFL Canvas approach.

The synchronization delay of the SL Canvas increases in proportion to the number of Canvas objects because the Canvas redraws all objects to update one object. There was little synchronization delay in the result of evaluation, however the Canvas has probably

Copyright: the authors

much more objects in collaborative web applications. For example, our collaborative paper editing support system draws a PDF document in Canvas with PDF.js. A Canvas rendered PDF document consists of many Canvas objects. Another example is WebGL, which renders a 3D graphics on Canvas. Therefore we should consider how to redraw Canvas objects to avoid loss of performance.

The length of the synchronization delay of the FL Canvas jumps when the number of objects increases. The reason is the size of Canvases. The size of Canvases increases dramatically in the FL Canvas. The FL Canvas minimizes the cost to redraw a Canvas because the Canvas redraws a necessary object only. Therefore we should consider the trade-off the size of Canvases and the cost to redraw a Canvas.

The synchronization delay of the DFL Canvas shows that the DFL Canvas is fast approach enough for Canvas object synchronizations. The synchronization delay is less than ten milliseconds even if a Canvas object has 500 or more objects. The result of the evaluation indicates that the DFL Canvas extracts frequently drawn objects only, and avoids increasing the size of memory usage.

6. Discussion

6.1. Number of Layers on DFL Canvas

Our research question is that how many layers an optimal number is on the DFL Canvas. The DFL Canvas has three types of layers: the real-time, the mutable and the immutable layer, however is not defined the number of each layer. The number of the layers is application-dependent because a synchronization delay of redrawing a Canvas is strongly affected by the number of Canvas objects in each layer.

The optimal number of layers in our collaborative paper editing support system is three. We target at a small group in the system such as members of a laboratory office, participants of a meeting and so on. Moreover, users of the system cannot add complex Canvas objects such as images. Therefore, the number of Canvas objects in our system is low enough for three layers. For instance, professors and students use our web application to editing their papers. There were four students and four papers. Three professors corrected these papers. Papers only include a 300 characters abstract. An average of 15.7 annotations per paper were added.

We should increase a mutable layer if a system with the DFL Canvas is acceptable huge number of Canvas objects. According to the evaluation, the synchronization delay is proportional to the number of objects in just one layer. Therefore, it should be considered to divide objects in a mutable layer.

6.2. Pattern in Paper Editing

We recognize a pattern in paper editing on our collaborative paper editing support system. Fig. 8, Fig. 9 and Fig. 10 show an editing pattern. This pattern was often seen in a paper editing.

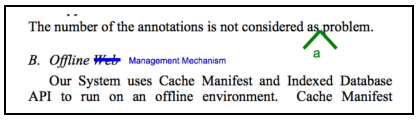

Fig. 8. Editing and inserting sentences.

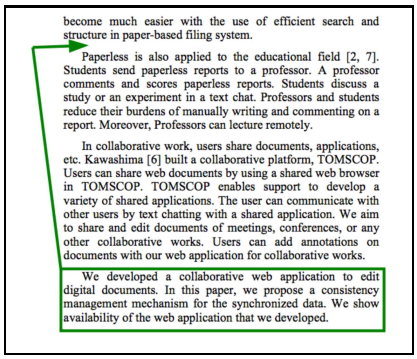

Fig. 9. Paragraph movement.

Copyright: the authors

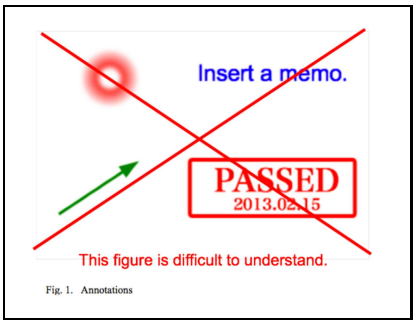

Fig. 10. Figure correction.

Fig. 8 shows a sentence editing and insertion. Professors use the line drawing to edit an incorrect sentence. The line was put on the vertical middle of the sentence. Professors present an edited sentence by adding the memo annotation. Fig. 9 shows a movement of paragraph. The professor instructs the student to move a lower paragraph to an upper position on the paper. Here, the professor used the square and arrow annotation. Fig. 10 shows a figure correction. The professor instructs the student to modify the figure. Here, the professor used the cross-mark annotation to instruct the student to change the picture.

7. Conclusion

We proposed the DFL Canvas approach for a fast synchronization. We implemented a collaborative paper editing support system with the DFL Canvas. We presented the evaluation result that indicates the approach is suited for collaborative paper editing support systems.

The DFL Canvas approach can realize effective features on collaborative paper editing support systems, e.g. the pointer that we described in this paper. Users can share their focuses using the pointer to collaborate each other. On the normal Canvas, which is called SL in this paper, the synchronization of Canvas objects is slow because Canvas redraws all objects. Users on collaborative paper editing support systems will want to share their focuses in order to collaborate each other when the conflict of their editing is occurred. The DFL Canvas approach enables to synchronize Canvas objects fast enough for paper editing.

References

1. S. Katayama, T. Goda, S. Shiramatsu, T. Ozono and T. Shintani, "A fast synchronization mechanism for collaborative web applications based on HTML5," *Proc. SNPD 2013*, **14**, 663–668 (2013).
2. D. Tilwaldi, T. Takahashi, Y. Mishima, J. Sawamoto, and H. Koizumi, "A smart method of cooperative learning including distant lectures and its experimental evaluations," *Proc. Intl. Conf. on Embedded and Ubiquitous Computing*, 268–277 (2005).
3. X. Li, S. Nakagawa, Y. Muramatu, and S. Sakaki, "Web-based collaborative correction support system for experiment report," *IPSJ SIGCE Technical Reports*, **2004**(100), 1–8 (2004).
4. D. Sun and C. Sun, "Context-based operational transformation in distributed collaborative editing systems," *IEEE Trans. on Parallel and Distributed Systems*, **20**(10), 1454–1470 (2009).
5. M. Heinrich, F. Grüneberger, T. Springer and M. Gaedke, "Exploiting annotations for the rapid development of collaborative web applications," *Proc. the 22nd Intl. Conf. on WWW*, **22**, 551–560 (2013).
6. M. Goldman, G. Little and R. Miller, "Real-time collaborative coding in a web IDE," *Proc. the 24th annual ACM symposium on User interface software and technology*, **24**, 155–164 (2011).
7. M. Heinrich, F. Lehmann, T. Springer and M. Gaedke, "Exploiting single-user web applications for shared editing: a generic transformation approach," *Proc. the 21st Intl. Conf. on WWW*, **21**, 1057–1066 (2012).
8. T. Ozono, R. M. E. Swezey, S. Shiramatsu, T. Shintani, R. Inoue, Y. Kato and T. Goda, "A real-time collaborative web page editing system WFE-S based on cloud computing environment," *Proc. IIAI Intl. Conf. on Advanced Applied Informatics*, 224–229 (2012).
9. R. Inoue, Y. Kato, T. Goda, T. Ozono, S. Shiramatsu and T. Shintani, "A real-time collaborative mechanism for editing a web page and its applications," *Proc. Intl. Symposium on Parallel Architectures, Algorithms and Programming*, **5**, 186–193 (2012).
10. T. Kawashima, and J. Ma, "TOMSCOP a synchronous P2P collaboration platform over JXTA," *Proc. 24th Intl. Conf. on Distributed Computing Systems Workshops*, **24**, 85–90 (2004).
11. H. Sano, S. Shiramatsu, T. Ozono and T. Shintani, "Implementing a web annotation system for supporting cooperative works using tablet devices," *Intl. J. Computer Science and Communication Networks*, **3**(1), 21–28 (2013).

Copyright: the authors

12. R. Suzuki, T. Doi, S. Shiramatsu, T. Ozono, and T. Shintani, "Implementing a synchronization mechanism of a pointer on meeting support system," *Proc. IEICE General Conf. on Information and System*, (1), 104 (2011).
13. C. Sols, J. H. Cans, and M. R. S. Borges, "Multilayer superimposed information for collaborative annotation in wikis," *Proc. Intl. Conf. on the Move to Meaningful Internet Systems*, **1**, 340–357 (2010).
14. J.-J. Lo, Y.-C. Wang and S.-W. Yeh, "Development of a synchronous collaborative writing revision instrument for teaching English," *Proc. Intl. Conf. on Information Management and Engineering*, 128–132 (2009).
15. H. Song, Y. Qi, Z. Ou, Y. Hu, Z. Zhang, and S. Ye, "A general collaborative editing platform based on file locking mechanism," *Proc. IEEE Intl. Conf. on Computer Science and Automation Engineering*, **2**, 436–440 (2011).

Copyright: the authors