What's Next? A Visual Editor for Correcting Reading Order

Daisuke Sato, Masatomo Kobayashi, Hironobu Takagi and Chieko Asakawa

Tokyo Research Laboratory, IBM Japan, 1623-14, Shimo-tsuruma,
Yamato City, Kanagawa, Japan, 242-8502
{dsato, mstm, takagih, chie}@jp.ibm.com

Abstract. The reading order, i.e. the serialized form, of the webpage should be a meaningful order for alternative representations such as the audible forms needed for visually impaired users. However, the serialized form rarely receives attention because it is visually elusive for authors using the existing WISIWYG authoring environments. Therefore we propose a new visualization technique called "reading flow" that visualizes the order of the serialized form with variable granularity by using a visible path extending through the elements in the content. This allows the authors to instantly evaluate the ordering by the visual pattern of the path. Our approach also allows them to interactively and intuitively reorganize the order of the serialized form. The results of two comparative experiments show that our reading flow greatly increases the ability of the authors to understand and organize the ordering compared to the existing techniques.

Keywords. Reading flow, reading order, Web accessibility, ARIA flowto

1 Introduction

With advances in Web technology, there are increasing demands for highly visual webpages. At the same time, it's extremely helpful if the serialized form of each webpage appears in a meaningful order for alternative representations, such as the audible forms used by visually impaired users or the transcoded forms used on small devices. For example, when a set of lists and headings for the lists are arranged in multiple columns in a table, all of the headings are read first, and then each of the lists is read sequentially (see Figure 1). Each list is separated from its title, and the intended semantics are hidden in the audible form. This is a well known, common, and severe accessibility problem. Authors should make the order meaningful at authoring time. This is mentioned in major accessibility guidelines and regulations. For example, the W3C Web Content Accessibility Guidelines (WCAG) 1.0 [1] rate the reading order adjustments at conformance level "A", the highest priority. Section 508 of the U.S. Rehabilitation Act [2], JIS X 8341-3 in Japan [3], and the eEurope Action Plan 2002 [4] all mention reading order.

However, the order of the serialized form (which reflects the order in the source code) is rarely noticed, because the existing WYSIWYG authoring tools allow authors to edit the content without considering the underlying source code order. The

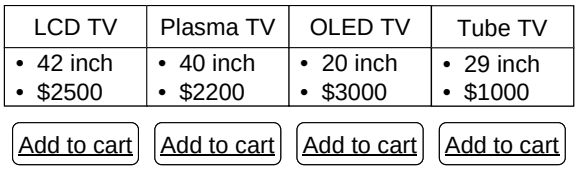


Fig. 1. An example of an inaccessible layout using a table. This would be read as "LCD TV, Plasma TV, OLED TV, Tube TV, 42 inch, $2500, 40 inch, $2200, 20 inch, $3000, 29 inch, $1000, link add to cart, link add to cart, link add to cart, link add to cart."

authors would have to modify the source code directly to control the ordering. Although there are many types of accessibility checking tools, the order of the serialized forms is rarely evaluated automatically because of the essential difficulty of algorithmically defining an appropriate reading order. Existing tools force authors to manually inspect the order of the serialized forms, but this is not intuitive and the tools do not support changing the order.

Therefore we propose a new visualization technique called "reading flow" that visualizes the order of the serialized form by using a visible path extended through the elements in the content. The reading flow represents the visual flow of reading the corresponding elements, and can represent fine or coarse granularity of the flow of the content. This allows an author to immediately see the serialized form and easily adjust it. The technique reduces the work for authors and developers when they build the webpages, since they can freely modify the reading order separately from the page design tasks. Going beyond webpages, our technique can potentially be applied to other types of documents where the reading order causes problems, such as presentation documents, PDF documents, and Flash content, and can help improve their accessibility.

The rest of this paper is organized as follows: Section 2 describes related work, Section 3 presents the proposed reading flow technique, Section 4 describes our implementation of reading flow, Section 5 reports on user experiments with reading flow, and Section 6 discusses the results of the experiments and some residual problems, and then concludes this paper.

2 Related Work

2.1 Transcoding

One major approach that can control the reading order without modifying the original content involves transcoding on a proxy server [5, 6]. This modifies the original content en route to a browser by using predefined metadata that can adjust the priority of each part of the content for blind users. There are also client-side techniques without proxies that generate alternative user interfaces for blind users by using predefined metadata [7, 8]. These techniques can also exploit a visual metadata editor but there is no interface to visually organize the reading order. The HearSay browser analyzes the content and the history of the link navigation on the fly to assess the

most important parts of the content for the blind users, and then reorders the content by considering each part's importance to the users [9]. One of the interesting flexible aspects is that the resulting reading order may change depending on the links that access the page, since those links may reflect different purposes and thus importance. All of these techniques could use our reading flow metadata when modifying or reordering the content.

2.2 Accessibility Checking Tools

There are various types of automated tools to check whether or not a document complies with the accessibility standards such as WCAG [1]. Some of them also have functions to simulate or visualize the serialized forms of the content.

Textualization is the basic function provided by WAVE [10, 11], aDesigner [12], and many other tools. The textualizing functions provide text-only views that simulate the serialized forms of the content. However, this function does not offer an intuitive way for users to evaluate the appropriateness of the content order because it loses the positional information of the display. The users must struggle to understand the relationships between the text and the original visual layout.

Numbering is an advanced function provided by WAVE. It visualizes the content order by showing the sequence as a number with each element. Although the number is clear about the order of the content, it is still hard to get an overview of the reading order. The aDesigner tool also provides a function to visualize the content order using gradations of the background color, thus showing the reaching time from the top of the webpage in the screen readers, but there is no easy way for users to determine which elements come first in the content. These tools are mainly used for checking the accessibility, and they don't have functions to organize the order of the content.

2.3 Tools for Optical Character Recognition (OCR)

In the field of OCR, there are also problems with the reading order of the content resulting in a lot of research. Various technologies predict the reading order based on layout analysis [13] or machine learning techniques [14]. Those automatic predictions may not be perfect. Therefore tools to visualize the results and to allow users to correct the results are provided. Such tools also include functions that allow users to adjust the reading order, which is important when validating OCR results [15, 16, and 17]. One system visualized the order between pairs of recognized text zones in the captured image by using arrows. Our reading flow was motivated by these techniques. We have extended the functions to handle the specific accessibility problems of HTML documents, and also improved the usability and the presentation.

3 Reading Flow

Our reading flow can be regarded as an interaction technique to obtain and organize the order of the serialized forms of the webpages. This section describes three major

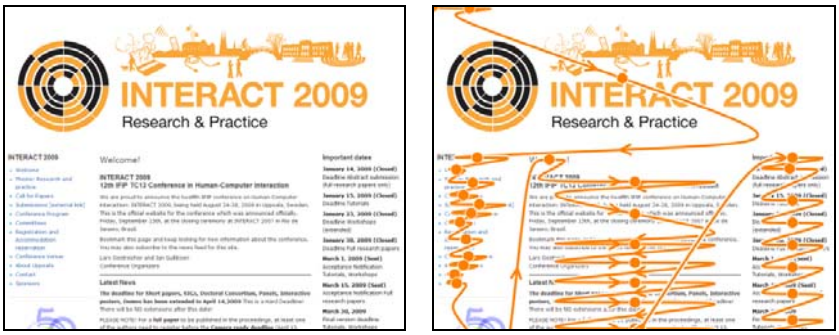


Fig. 2. A fine-grain visualization of the reading flow of the INTERACT 2009 webpage. Each distinguished element has a knob (shown as a circle) on the path of the reading flow

Table 1. The relationships between the properties of elements and their visualization.

	Audible		Not audible
	On the screen	**Off the screen**	**On and off the screen**
Visible	Displayed	Collapsed	Not Displayed
Hidden	Collapsed	Collapsed	

features of the reading flow: visualization, authoring, and granularity control of the reading flow.

3.1 Visualization of Reading Flow

The reading flow visualizes the order of the serialized forms of the webpages by using a traversal path aligned with the elements of the content. Figure 2 shows an example of the visualization of our reading flow on the webpage of INTERACT 2009. Our reading flow will be displayed over the rendered image of the content, which can be scaled to any size. The reading flow is represented by smoothly connected arrows that indicate the visual flow of reading the corresponding elements. The visual flow of reading each element is determined by its content and the mode of the characters, which can be left-to-right and top-to-bottom, top-to-bottom and right-to-left or some other order. Not all of the elements need to be included on the path. Multi-granularity visualization is an extension of this approach (Section 3.3). In the finest-grain visualization, three properties of the element are considered to support skipping unnecessary elements. First, whether or not the element is audible for reading by screen readers. Second, whether the element is visible or hidden. Third, whether or not the element is located in the visible part of the screen (see Table 1). The traversable path of the reading flow must pass through all audible, visible, and on-screen elements. The parts of the path for the hidden elements and the visible but off-screen elements can be collapsed and displayed as special tags at appropriate positions on the complete path. By clicking the tags, the corresponding collapsed paths can be displayed.

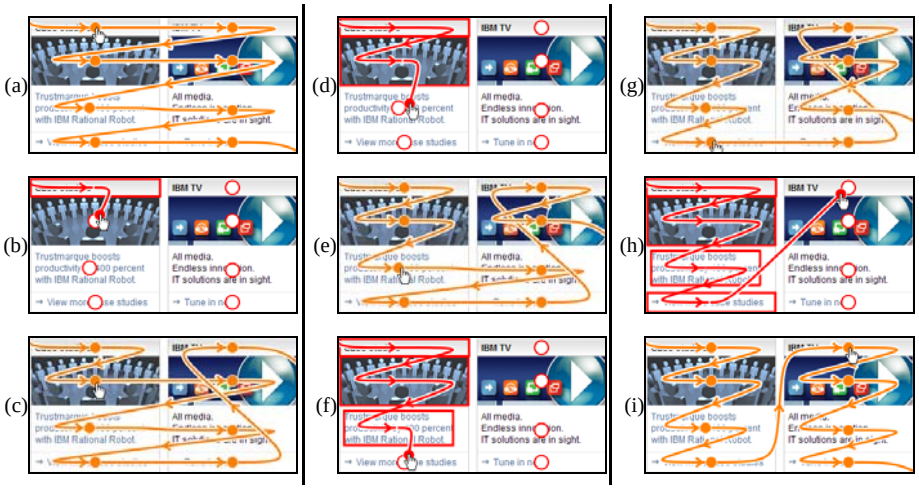


Fig. 3. An example of organizing the reading flow. Part (a) shows the original reading flow of the content, (b), (d), (f), and (h) show snapshots while the dragging the knobs. By dragging the first knob at the upper left to the knob below it (b), the skipped knob at the upper right is moved to the end of the path (c). The pairs (d) – (e), (f) – (g), and (h) – (i) show similar reordering operations to create a two-column layout in this example.

3.2 Authoring of Reading Flow

The reading flow also provides a knob on each element so that the author can explicitly and visually change the ordering by connecting each knob to the next one. Figure 3 shows an example of reordering. If a user finds an incorrect order on the reading flow at an element (Figure 3a), the user can easily change the flow by dragging the knob on that element to the knob of the proper next element. During the dragging operation, the prior path of the dragged knob is highlighted and the rest of the path is temporally hidden (Figure 3b and 3c). The interface uses three constraints to maintain the consistency of the reading order in spite of the reordering and to make the interface as simple as possible (so even novice users can use it). (1) A knob can only be connected to another knob going forward. In other words, the user connects a knob to some later element or an unlinked element, not to an earlier element. (2) A knob cannot be connected to make a loop in the reordered path. (3) If a group of elements becomes disconnected, then that segment is moved to the end of the current path.

3.3 Granularity Control of Reading Flow

Our tool has a slider that allows users to determine the granularity level of the reading flow. Figure 4 shows an example of controlling the granularity of the reading flow. With the fine-grain reading flow the users can check the path to each element in the content (Figure 4a). By selecting a higher granularity level, the users can grasp the

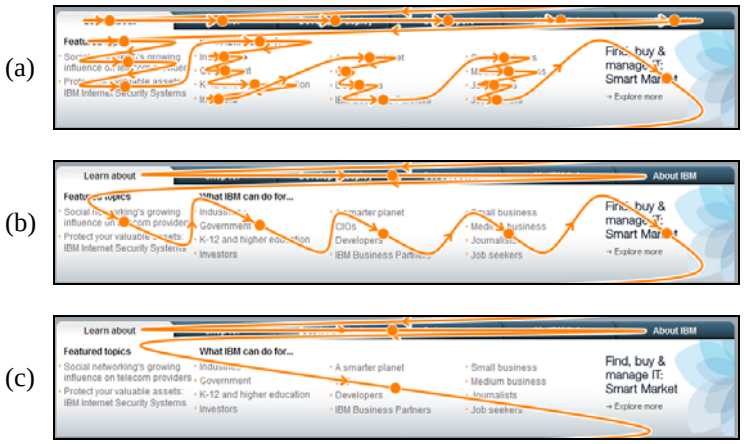


Fig. 4. (a) Shows the most details of the reading flow, (b) shows a medium abstraction level, and (c) shows the highest level of abstraction of the reading flow.

overview quality of the reading order in the page. For example, the items of a list now have a left-to-right flow and are aligned at the left side of the list (Figure 4b), and at the highest level, the entire series of lists is regarded as one object (Figure 4c). The reordering in a granularity level affects the other levels of granularity, so that the users can edit the order of the flow at any granularity level that is displayed.

The granularity control is implemented as a grouping of elements by using geometric similarities in the tree structure of the content, because the layout of the content is often separated from the tree structure by the style sheets. The system calculates the linkability between adjacent elements in the flow, and then assembles the pairs of elements having highest linkability into a group. The linkability is determined by the geometric similarity and the size of the elements, and the more similar and smaller elements get the higher linkability scores. The system continues this step of prioritizing elements until the number of elements is greater than a target number determined by the setting of the slider for the granularity control. In this process certain pairs are excluded, such as a pair of widely separated elements or a pair that causes an unnatural flow (as when the second element of a pair is above the other element within a top-to-bottom block of text).

4 Implementation

Our reading flow is currently implemented as a function in a Firefox extension written in JavaScript. The metadata generated by the tool is in the Accessibility Commons format [18] and can be shared through the Web service as XML or JSON.

The current implementation handles the order of the content in a semantic layer separated from the original content to reduce the complexity and to allow for sharing the order information as accessibility metadata. When a user modifies an order, the

modification is stored as a set of metadata to change the reading. The metadata is created with a "flowto" concept, which is defined in the working draft of the Accessible Rich Internet Applications (WAI-ARIA) version 1.0 [19]. Each flowto attribute has an ID for a target element to be read next.

It is easy to add flowto attributes to the content, but currently (January 2009) there are no screen readers that can directly use the flowto attributes to control the order of the content. Therefore currently the flowto metadata is applied to the content inside browser on the client side to simply change the order in the DOM tree of the original content, and ignoring possible disruptions of the appearance. The flowto metadata also can be converted into metadata for other systems that seek to change the reading order, such as transcoding systems.

5 Experiments

We carried out a two-task experiment to empirically evaluate user performance using our reading flow technique. The first task involved finding problems in the reading order. The second task studied the editing process for the reading order. In the experiment we compared our technique with a baseline numerical sequence display, controlled by entering sequence numbers into the interface.

5.1 Participants

Twelve computer science researchers (9 males and 3 females) from 25 to 33 years of age participated in the experiment. (One participant's age was unknown.) All had experience with graphical user interfaces. Seven had professional or partial knowledge of accessibility. Four had prior knowledge of the reading order problem. We made a selection of the participants and focused on their expertise in accessibility, because the target users of these user interfaces are people who want to correct accessibility problems, such as Web designers and Internet volunteers.

5.2 Apparatus

We used a 3.8 GHz Xeon workstation running Windows XP, connected to a 17-inch LCD display with a resolution of 1280 × 1024 pixels and a standard optical mouse. In the tests involving our reading flow system, the reading order was visualized as arrows displayed on top of the target webpage and modified by the mouse operations described in Section 3.

In the numerical interface, the positions were presented as sequence numbers on each element in the page. Each number was displayed using black digits in a small white rectangle. An input dialog for modifying the number popped up when the user clicked on a button next to the number. If the user entered an existing number, the other elements with that number and the following numbers were automatically incremented so that each element always had a unique number. All of the target

webpages in the experiment were small webpages that could be displayed in the experimental window without scrolling.

5.3 Task 1: Determining the Quality of the Reading Order

Each accessibility trial involved five steps: (1) Before beginning the trial, each participant saw a screenshot of the target webpage. The participant was allowed to study the structure of the page until satisfied. (2) The participant clicked anywhere on the screen to start the timed portion of the actual trial. (3) The reading order was displayed on top of the page. (4) The participant clicked on the screen again after deciding whether or not the reading order of the target webpage was accessible. (5) The participant then recorded the decision by clicking on a good or bad button. The participants were instructed to complete each trial as quickly as possible.

We measured task completion time as the time between Steps 2 and 4 and the error rate was defined as the percentage of trials in which participants failed to correctly assess the quality of the reading order.

We used a within-participant design. The independent variables were the interface (fine-grain reading flow, coarse-grain reading flow, or sequence number display), reading order quality (good, somewhat problematic, or very problematic), and target webpage (e-commerce or airport). We tested 18 combinations in total. Each participant performed one trial for each combination. The presentation order of the interfaces was counterbalanced. The quality of the reading order and the target webpage were presented in a random order.

To introduce this task, we instructed participants about accessible and inaccessible reading orders, including several good and bad examples. The participants also had a training trial for each interface to become familiar with the task. Each session took approximately 10 minutes, including the training.

5.4 Task 2: Revising the Reading Order of Contents

Each editing trial involved four steps: (1) Before beginning the trial, each participant saw two screenshots, one with the initial reading order and another with the desired reading order. The participant was also told about the specific problems of the initial order and given instructions how to fix the problems. (2) The participant clicked on a start button to begin the trial. (3) The participant modified the reading order to change it to the desired order using one of the two interfaces. (4) The trial ended as soon as the modified reading order matched the desired order. During this task, the granularity slider was set to “fine” because several steps in this task required low-level changes in the reading order, which could only be performed with the fine-grain arrows. The participants were asked to complete each trial as quickly as possible. We defined the task completion time as the time between Steps 2 and 4. We did not measure error rates because each trial continued until the reading order matched the desired order.

We again used a within-participant design. The independent variables were the interface (reading flow or sequence number entry), reading order quality (somewhat problematic or very problematic), and target webpage (e-commerce or airport). We

tested eight combinations in total. Each participant performed one trial for each combination. Half of the participants had four trials of the reading flow interface first, followed by four trials of the number sequence entry interface. The other half of the participants had the reverse order. The quality of the reading order and the target webpage were presented in a random order.

At the beginning of each session, both interfaces were described. The participants also received a training trial for each interface to familiarize them with this task. Each series of sessions took approximately 20 minutes, including training. After finishing all of the sessions, the participants filled in a questionnaire that covered both tasks.

5.5 Results

Figure 5a shows the task completion times for each reading order quality and target webpage combination in Task 1. In this data, we excluded all of the trials with errors. The average values were 6.63 seconds for fine-grain reading flows, 4.99 seconds for coarse-grain reading flows, and 10.18 seconds for numerical displays. The numerical display interface took 53% longer than the fine-grain reading flow and 104% longer than the coarse-grain reading flows. Analysis of variance showed significant primary effects of the interface ($F_{2,178} = 36.69$, $p < .001$), reading order quality ($F_{2,178} = 14.55$, $p < .001$), and target webpage ($F_{1,178} = 8.529$, $p < .005$). There were interaction effects for the interface × reading order quality. A post hoc analysis indicated that the coarse reading flow is significantly faster than the fine reading flow ($p < .05$) and reading flows are significantly faster then the sequence number display ($p < .001$). It also indicated that the good reading order took a significantly longer time to confirm than to detect problematic conditions ($p < .001$). There was no significant difference between the degrees of problems.

Figure 5b shows the incorrect assessment rate for each combination of reading order quality and target webpage in Task 1. The overall values were 4.2% for each interface. Most errors involved the less problematic conditions. Analysis of variance showed a significant primary effect of the reading order quality on the error rate ($F_{2,187} = 6.872$, $p < .005$). The interface and target website had no significant effects, and there were no interaction effects. A post hoc analysis indicated that somewhat problematic conditions caused significantly more errors than good or very problematic conditions.

Figure 6 shows the task completion time for each combination of reading order quality and target webpage in Task 2. The overall average values were 31.6 seconds for reading flows and 49.1 seconds for numerical entries. The numerical entry interface took 56% longer than the reading flow interface. The reading flow interface generally outperformed the numerical entry interface except for the e-commerce website in the somewhat problematic condition. Analysis of variance showed a significant primary effect of the interface on this value ($F_{1,77} = 9.226$, $p < .005$). The reading order quality and target website had no significant effects and there were no interaction effects.

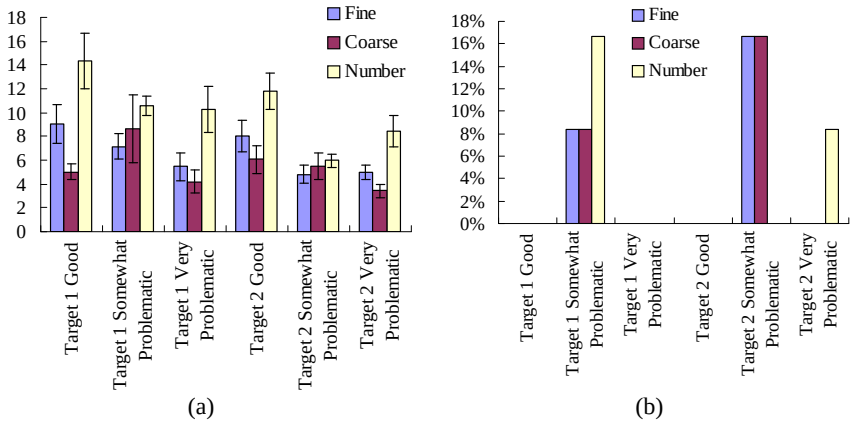


Fig. 5. (a) Task completion times (seconds) with standard errors and (b) error rate (%) for quality assessment.

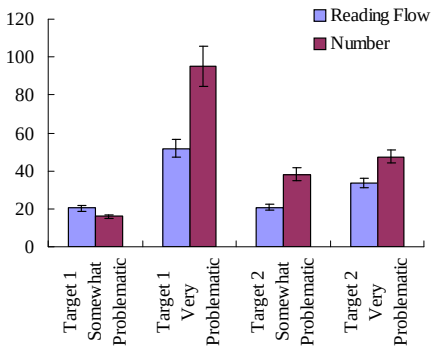


Fig. 6. Task completion times (seconds) with standard errors for reading order revising.

The post experiment questionnaire asked for subjective evaluations of our reading flow interface. All eleven participants preferred the reading flow interface to the numerical display interface for both tasks. For Task 1, six participants preferred coarse-grain reading flows, two preferred fine-grain reading flows, and the others had no preference between the reading flows granularities.

6 Discussion

6.1 Effectiveness and Visual Patterns

The results clearly show the effectiveness of this new visualization method. Subjects could perform the tasks in significantly shorter times (Figure 5a) by using the visualization than by using the sequence number display. Also, the subjects preferred the visualization over the sequence number display (Figure 6). The coarse-grain

visualization also accelerated the performance. In most cases, coarse-grain visualizations achieved significantly faster performance than fine-grain visualizations (Figure 5a), even though the error rates were the same (Figure 5b).

The differences in task completion time were significant between visualization and sequence number display in all cases, but the degree varied depending on the quality of each target page. When a page did not have any problems ("good" in Figure 5a) or when a page had many problems ("very problematic"), the difference is much larger than for pages with a smaller numbers of problems ("somewhat problematic").

One possible interpretation of the result is that subjects could recognize visual patterns for good or very problematic pages at a glance. The visual patterns are not clear from the interview comments, but we think that one of cues is the number of crossing points in the visualization. The line traverses the content with fewer crossing points if the reading flow matches smoothly with the visual semantics, otherwise the line becomes tangled with many crossing points (Figure 7a). In contrast, for "somewhat problematic" pages, the problem is elusive in the smooth reading flow line, which is not visually obvious in comparison to good pages (Figure 7b). The error rate was significantly higher in those tasks, even though subjects spent substantial time with somewhat problematic pages (Figure 5b). This may indicate that subjects overlooked small visual patterns with problems.

If this hypothesis is true, it should be possible to improve the visualization by highlighting the characteristic visual patterns of problems on a page. For example, the color gradation for each segment of the flow line from the top to the end of a page could improve the visibility of visual patterns by giving a sense of the distance for each intersection. A more direct enhancement might be the highlighting of each crossing point with a red flag on the background. If a visualization is filled with red, then the page may have many problems. Another type of improvement involves few crossing points. For example, Figure 7b looks like a good example with a smooth line and no crossing points, but it has the table layout problem. This example resulted in the highest error ratio (Figure 5b "Target 2 Somewhat Problematic"). This type of problem can be easily recognized by comparing the reading order with the original content without considering the visual patterns. Therefore, it is necessary to warn about potential problems in the visualization. For example, if knobs have different colors for table elements, that could warn users to carefully compare the semantic reading order with visualization.

6.2 Interface for Correction

The result of Task 2 shows the significance of the visualization for correction tasks compared with the numerical interface. However, the difference is not clear in some cases. For example, the somewhat problematic example for target 1 had results that were reversed from the other cases (Figure 6). Subjects commented that they consistently tried to scan the numbers from the top of a page, and changed the values to satisfy the semantic ordering. For the reversed results, the page can be fixed by changing only a few numbers close to the top of the page. Then the subjects could finish more quickly, since the time duration for each repair operation is shorter than the visualization. One subject also commented that the temporary intermediate

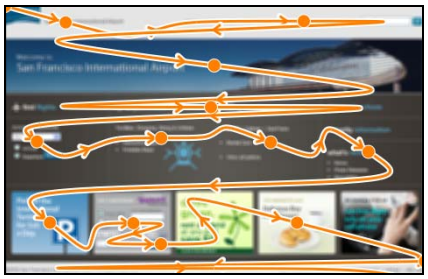
(a)  (b) 

Fig. 7. Examples of reading flows with problems. (a) Is very problematic, and (b) has a problem related to table heading order, and is somewhat problematic.

visualizations during a sequence of repair operations were often hard to interpret as the lines became tangled.

These results give insights into the characteristics of the repair operations. The visualization was very quick and effective when subjects could recognize the problems in a page, and the improved recognition times also contributed to the improvements in the correction task (Task 2). However, the correction operation based on mouse dragging was relatively less time effective and also the method for temporary intermediate visualization was not completely effective for the subjects during the task.

Several areas for possible improvements appear in these results. The biggest area might be a combination interface with both sequence numbers and the visualization. The numbering of each element would be the simplest combination and also temporary and local numbering may improve the usability when changing the order of a series of elements. The number-based correction also can be integrated into the visualization along with the mouse-based interface that uses dragging and clicking. We will consider these improvements as future work.

6.3 Possibility for Collaborative Correction

It is clear from the experiment that the visualization allows users to correct reading order significantly faster. If the method becomes sufficiently easy for average Web users, collaborative accessibility improvements for ordering by community volunteers will become possible in the near future. This would be part of a new strategy to make a wide range of Web content accessible by gathering the power of Web users, based on the same social computing techniques as used in the widely accepted wikis and social bookmarking services. Projects by Takagi et al. [20] and Jeffly et al. [21] are leading projects in this area, allowing volunteers to fix various accessibility problems, such as missing alternative texts and missing heading tags. Currently, the services do not have any function to allow volunteers to reorganize reading order, mainly because the existing reordering interfaces (e.g. sequence number display) are far too difficult for typical Web users to understand without special training. The reading flow will be one of the key technologies for such services to provide ideally accessible webpages for the blind. We plan to deploy these functions to help blind users who face difficulties in the current visual Web environment.

6.4 Feedback to the Original Content

Although the metadata for reading order is completely separated from the original content in our current implementation, there is a various possibilities for applying the metadata to improve the content. The metadata can be used to modify the source of the content or to change the live DOM on the client browsers, but the feedback system for the reading order without affecting its appearance and the usability is a challenging problem because the structure of the content and the style sheets influence the appearance and the usability. However this is a final piece when trying to realize a full ecosystem of checking and organizing the reading order for the collaborative corrections that help the blind users.

7 Conclusion

A new method to visualize the reading order of webpages was introduced. Our "reading flow" approach allows Web authors to easily and intuitively evaluate the appropriateness of the reading order or their webpages. We have shown how the reading flow visually exposes the order of webpages with a variable granularity, supporting functions for visually reordering the elements, and generates reading order metadata that is separate from the original content. Two sets of user experiments were conducted comparing the performance in recognition and the performance of repair tasks in the reading order (against a numerical interface). The results show that the reading flow technique is significantly faster than a numerical interface. The result also suggested several areas for improvement, both in visualization and in the interface for repairs.

As future work, we are planning to implement improvements and investigate feedback mechanisms for applying the reading order metadata inside the original content and the effectiveness for the blind users. The visualization will be a key technology to allow average Web users to work on improving Web accessibility in the social approach. We hope to deploy the visualization function to broader audiences, and to provide the benefits of this new metadata for correcting reading order for blind users worldwide.

References

1. Web content accessibility guidelines (WCAG) 1.0. W3C Recommendation 5 May 1999, http://www.w3.org/TR/WCAG10/.
2. Section 508 of the rehabilitation act. http://www.section508.gov/.
3. Japanese Standard Association. JIS X 8341-3:2004 guideline for older persons and persons with disabilities -information and communications equipment, software and services-part3 Web content.
4. Council of the European Union. eEurope 2002 -an information society for all -action plan
5. Hironobu Takagi, Chieko Asakawa, Kentarou Fukuda, and Junji Maeda. Site-wide annotation: reconstructing existing pages to be accessible. In ASSETS '02: Proceedings of the Fifth international ACM conference on Assistive technologies, pages 81–88, 2002.

6. Simon Harper and Sean Bechhofer. Sadie: Structural semantics for accessibility and device independence. ACM Trans. Comput.-Hum. Interact., 14(2):10, 2007.
7. Hisashi Miyashita, Daisuke Sato, Hironobu Takagi, and Chieko Asakawa. Aibrowser for multimedia: introducing multimedia content accessibility for visually impaired users. In ASSETS '07: Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility, pages 91– 98, 2007.
8. Christos Kouroupetroglou, Michail Salampasis, and Athanasios Manitsaris. A semantic-Web-based framework for developing applications to improve accessibility in the www. In W4A: Proceedings of the 2006 international cross-disciplinary workshop on Web accessibility (W4A), pages 98–108, 2006.
9. I. V. Ramakrishnan, Amanda Stent, and Guizhen Yang. HearSay: enabling audio browsing on hypertext content. In WWW '04: Proceedings of the 13th international conference on World Wide Web, pages 80–89, 2004.
10. WAVE 4.0 accessibility tool. http://wave.webaim.org/.
11. Leonard R. Kasday. A tool to evaluate universal Web accessibility. In CUU '00: Proceedings on the 2000 conference on Universal Usability, pages 161– 162, 2000.
12. Hironobu Takagi, Chieko Asakawa, Kentarou Fukuda, and Junji Maeda. Accessibility designer: visualizing usability for the blind. In ASSETS '04: Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility, pages 177–184, 2004.
13. Jean-Luc Meunier. Optimized xy-cut for determining a page reading order. In ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition, pages 347–351, 2005.
14. D. Malerba, M. Ceci, and M. Berardi. Machine Learning for Reading Order Detection in Document Image Understanding. Machine Learning in Document Analysis and Recognition, 2008.
15. Sherif Yacoub, Vinay Saxena, and Sayeed Nusrulla Sami. Perfectdoc: A ground truthing environment for complex documents. In ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition, pages 452–457, 2005.
16. BA Yanikoglu and L. Vincent. Pink panther: a complete environment for ground-truthing and benchmarking document page segmentation. Pattern Recognition, 31(9):1191–1204, 1998.
17. T. Kanungo, C.H. Lee, J. Czorapinski, and I. Bella. TRUEVIZ: a groundtruth/metadata editing and visualizing toolkit for OCR. In Proc. SPIE Document Recognition and Retrieval VIII, volume 4307, pages 1–12.
18. Shinya Kawanaka, Yevgen Borodin, Jeffrey P. Bigham, Darren Lunn, Hironobu Takagi, and Chieko Asakawa. Accessibility commons: a metadata infrastructure for Web accessibility. In ASSETS '08: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility, pages 153–160, 2008.
19. Accessible rich internet applications (WAI-ARIA) version 1.0. W3C Working Draft 6 August 2008, http://www.w3.org/TR/wai-aria.
20. Hironobu Takagi, Shinya Kawanaka, Masatomo Kobayashi, Takashi Itoh, and Chieko Asakawa. Social accessibility: achieving accessibility through collaborative metadata authoring. In ASSETS '08: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility, pages 193–200, 2008.
21. Jeffrey P. Bigham and Richard E. Ladner. Accessmonkey: a collaborative scripting framework for Web users and developers. In W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A), pages 25–34, 2007.