Intra/Inter-document Change Awareness for Co-authoring of Web Sites

Stavroula Papadopoulou, Claudia-Lavinia Ignat, Gérald Oster, Moira C. Norrie

Intra/Inter-document Change Awareness for Co-authoring of Web Sites

Stavroula Papadopoulou[1], Claudia-Lavinia Ignat[2], Gérald Oster[2], and Moira C. Norrie[1]

[1] Department of Computer Science, ETH Zurich, Switzerland
{papadopoulou,norrie}@inf.ethz.ch,
[2] LORIA, INRIA Nancy-Grand Est, France
{ignatcla,oster}@loria.fr

Abstract. Systems that support the co-authoring of web sites often allow users to freely edit pages. This can result in semantic inconsistencies within and between pages. We propose a change awareness mechanism that monitors intra- and inter-document edits, taking into account changes made to a page and pages connected to it through *html* or *transclusion links*. The effect of all the changes is computed based on various metrics and on different semantic levels according to user preferences. A visualisation tool indicates how much a document and documents linked to it have changed. An edit profile allows users to easily spot parts with "interesting" changes within web pages.

1 Introduction

Until recently, the World Wide Web was used mainly as a read-only medium where authors publish to many readers. Now the trend is toward multiple interacting authors such as seen in the collaborative authoring of web sites and even web pages. Authors are often distributed in time and space, which can make the coordination of their work difficult. To avoid imposing certain strategies or working modes on authors, users are often allowed to freely edit web pages without constraints on when this happens or which parts are modified.

Changes to a web page may affect other parts of the page or pages linked to it and hence lead to semantic inconsistencies within or between pages. Keeping users aware of changes made by other users can help to resolve such inconsistencies or even prevent them. While many popular tools for the collaborative authoring of web pages such as Wikis, blogs and WebDAV applications succeed in enabling collaboration between users, they unfortunately provide almost no awareness to users about the changes done by their collaborators, and therefore do not prevent semantic inconsistencies.

In contrast, the Computer-Supported–Cooperative–Work (CSCW) community has identified awareness as an important feature of collaborative applications. *Change awareness*, defined as "the ability of a person to track the changes that other collaborators have made to a group project" [1], has been investigated

by many research teams dealing with collaborative applications for text [2, 3], graphical [1], or software engineering applications [4]. To the best of our knowledge, none of the existing CSCW approaches has been applied to provide awareness for the co–authoring of web pages where a web page can indirectly be modified through changes made to a linked page.

Inspired by the CSCW community, we address the need of users to track changes made over time to co–authored web pages. We identify two categories of changes that may appear: *intra–document changes* and *inter–document changes*. The first category includes changes made to a single web page. Users need to be informed in detail about the type of changes, which part of the page is affected, in which way and how much. An overview of the changes should be provided to the users so that they are aware of the group activity and can react quickly to changes that might affect their work. Web pages often contain links to other pages and the target pages may be semantically connected. Additionally, parts of web pages are often reused in other web pages resulting in "transcluded" web pages [5] indirectly linked to "compound" web pages [6] that include the reused information. In both cases, changes to a target/transcluded page may directly or indirectly influence the content of the source/compound page resulting in *inter–document changes*. In the process of collaborative authoring of a web site, it is therefore important that a user is also notified about concurrent changes made to linked documents.

Current solutions only offer the possibility to track changes made to a web page and therefore deal only with *intra–document* changes. While the possibility of following links from a source page is offered by some tracking tools, they do not relate the changes made to the target page with the source page. Further when *intra–document* changes are tracked, the awareness information presented to the user is computed at a single syntactic level of the document. Additionally, existing tools do not provide an overview of changes made or information about the user who made the changes.

In [7] we proposed a mechanism that computes awareness information about *intra–document* changes made to structured documents. We used this mechanism to extend an existing asynchronous collaborative text application to provide awareness information. We conducted a user study [8] to test our hypothesis that the resulting application provide increased awareness information and received positive feedback from the participants. We then applied the mechanism to a semi-synchronous collaborative text editor [9] and an asynchronous collaborative graphical editor [10]. All the editors used a structured document model. In this paper we show how we applied the same mechanism to collaborative situations where web sites are authored. We also show how we extended our awareness mechanism to deal with links between web pages.

We propose a mechanism that based on *html* and *transclusion links*, informs authors of source/compound web pages about changes made to the target/transcluded web pages. Our mechanism additionally computes and visualises change awareness about *intra–document* changes on the different syntactic document levels of one or more web pages. We begin in Section 2 with a motivating

example to show the kinds of information that would be useful for users when co–authoring web pages. In Section 3 we present our approach to computing change awareness for *intra–* and *inter–document* modifications and in Section 4 we introduce a visualisation tool to show the computed awareness information. We discuss related work in Section 5 and conclude in Section 6.

2 Motivating example

In this section we use an example to motivate the need for users to be informed about intra-document and inter-document changes to a web site. Consider the case of a research group that maintains a web site about their activity. The structure of the website is shown in Figure 1.

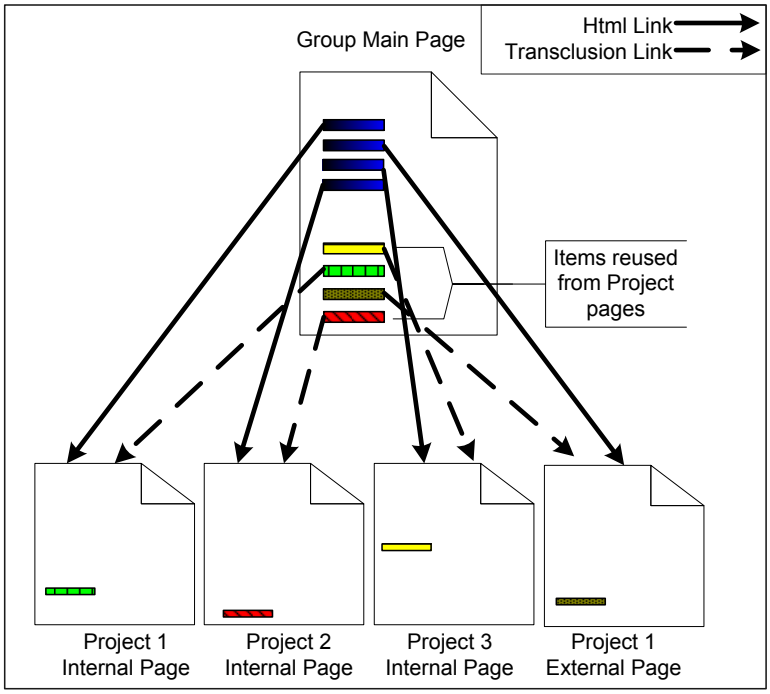

Fig. 1. Structure of a group web site.

The group is involved in several projects (Project 1, Project 2, Project 3) with a subset of group members participating in each. The group web site is composed of a main page and a set of project pages. The main page contains a description of the group research topics, a list of the group members, a summary of the group projects together with links to the internal web pages describing the projects, and links to an external web page of Project 1 maintained by

a collaborating research group. We assume that the summaries of the group projects are transclusions of the short descriptions of the projects found on the associated web pages. Figure 1 shows the *html* and *transclusion links* from the group main page to the projects internal and external pages. To keep the group web site consistent, users working on the main page would like to be continuously informed about changes made to:

- *The main page itself.* Users need to be informed in detail about the type and location of changes, as well as their effect on the parts of the page where they were made and the page as a whole.
- *The transcluded parts.* Any change made to the source document of a transcluded part must be tracked and the compound document must be notified. After refreshing the transcluded part to show its latest status, awareness information about the detailed changes made to it need to be given as well.
- *The linked pages.* Users editing the main page should be informed about changes made to the linked web pages as these changes might produce inconsistencies in the main page. For instance, if a new project member is added to an internal project, the complete list of participants on the group main site should be updated. In the same way, if a new software release is announced on the external page of Project 1, this information could be used to update the group main page.

The visualisation mechanism that presents change information to the users should provide an overview of changes as well as details at a selected document level such as a section or paragraph, the transcluded parts and the linked pages.

Some tracking tools allow the possibility to follow links from a source page, but they do not relate changes made to the linked pages with the source page. Therefore, a user working on the source page is not aware about changes made to the linked pages. In the following sections we present our awareness mechanism for intra and inter-document changes. Users working on a web page are informed about changes made to the main page and linked pages without visiting those web pages.

3 Computation of intra/inter–document awareness

In this section we present our awareness mechanism for intra- and inter–document changes. A flexible awareness mechanism is required to compute and visualise awareness information at a granularity level defined by the user, based on their current role and needs. We propose an awareness mechanism for documents conforming to a hierarchical structure. The use of an underlying structured document model enables the addressing of different document parts and document levels and therefore provides the required access to them which enables the tracking of changes made to them and the computation of the required awareness information. Our awareness mechanism includes the concepts of a node, i.e. a document part, an operation, i.e. a change made to a node, various metrics that can be used to measure the effect of an operation applied to a node

and visualisation tools to present the computed information. Finally, it includes the parent–child relationships between the document nodes that form the structured document. The relationships are used to compute the effect of a node's change to another node. All the above concepts are defined in a general way to allow the mechanism be applicable to any collaborative application that uses a structured document model, independently of the document type or the mode of collaboration.

We first explain the document model and the concept of operations used to describe web pages and the changes made to them, respectively. Then, we introduce the architecture of our approach and present in detail how we collect the intra- and inter–document changes made to web pages and compute the corresponding awareness information.

3.1 Document model and operations

We view each web page being collaboratively authored as a structured document. Through the structured document model, we can address different parts of a web page at various syntactic document levels and compute awareness information about how much they have changed at different granularity levels. We adopt the definition of a document node introduced in [11].

A node N of a document is a structure of the form
$N =<level, children, history, content>$, where

- *level* is the granularity level, $level \in \{0, 1, \ldots, n\}$ corresponding to node N,
- *children* is an ordered list $\{N_1 \ldots, N_m\}$ of child nodes,
- *history* is an ordered list of operations referring to child nodes,
- $content = \begin{cases} \text{object stored in node, if } N \text{ is a leaf node} \\ \sum_{i=1}^{n} content(child_i), \text{ otherwise} \end{cases}$

Changes to a web page are mapped to operations applied to specific parts of the page. Examples of operations are the insertion/deletion of a new paragraph to/from a section or of a sentence to/from a paragraph or of a word to/from a sentence etc. We define operations in a similar way to that described in [11].

An operation is a structure of the form op= <*type*, *level*, *position*, *content*, *length*, *user*>, where

- *type* is the type of the operation, $type \in \{insert, delete\}$,
- *level* is the level of the node in whose history the operation is kept,
- *position* is a vector of positions specifying the path from the root to the node where the operation is applied,
- *content* is a node representing the content of the operation,
- *length* is a vector with the number of units of each document level inserted or deleted by the operation,
- *user* is the user who generated the operation.

Following the document's structure, the effects of changes made to document parts at lower syntactic levels, such as word or sentence in the case of text,

are aggregated to show the total changes made to the document at a higher syntactic level such as a paragraph. Monitoring changes made to different parts of the document and evaluating their effect returns the required intra–document change awareness.

The operations that transform a web page from one version to another can be created in two ways: (1) by using a special web application, when authoring the page, that captures the changes and (2) by using a diff algorithm [12] between the two versions. When an application that captures user changes is used, the captured operations reflect the actual changes made to the page. When a diff algorithm is used, a set of operations that transform the initial state of the web page to its final state is created. Unfortunately, this set may not be identical to the set of actual changes made by users, resulting in loss of some information. The advantage, though, of a diff algorithm is that it can be used to compute the difference between two versions of any web page without monitoring the authoring procedure. Being aware of these issues, we chose to use a diff algorithm to enable the computation of awareness information for both internal and external pages. When addressing such problems, we believe that being able to offer awareness information, even if operations do not capture the exact changes, is far more beneficial than not delivering any information at all. Of course, in situations where only internal pages are authored and the accuracy of computed awareness is crucial, an authoring application that captures user changes can be used instead of a diff algorithm.

3.2 Architecture

The procedure followed to compute intra– and inter–document changes and the corresponding awareness information is shown in Figure 2. A *PageCache* is maintained locally for each user, including a copy of each of the web pages of interest to the user, as well as a copy of all the pages linked to them. We refer to all of these as *monitoredPage*s. Additional information is kept in each *monitoredPage* showing how much each part has changed. A background process periodically calls the method *updateValues* for each *monitoredPage* to check whether any intra– or inter–document changes occurred and compute their effect.

To compute awareness information for the intra–document changes, we call the method *getIntraValues* for the currently monitored page. *getIntraValues* takes the cached (old) version of the page and the online (current) version from the web and compares them. If they are not identical, the procedure in the block "condition", shown in Figure 2, is followed. Initially, a diff algorithm is used to compute the changes that transform the old version of the page into the current one in terms of operations denoted as *changes*. The *changes* are used by the method *computeIntraValues* to compute the *intraValues* attached to the changed parts of the page. This procedure is detailed in Subsection 3.3. Finally, these values are stored at the *monitoredPage*, the new version of the page is stored at *PageCache* and *getIntraValues* returns the computed intra values associated with parts of the document.

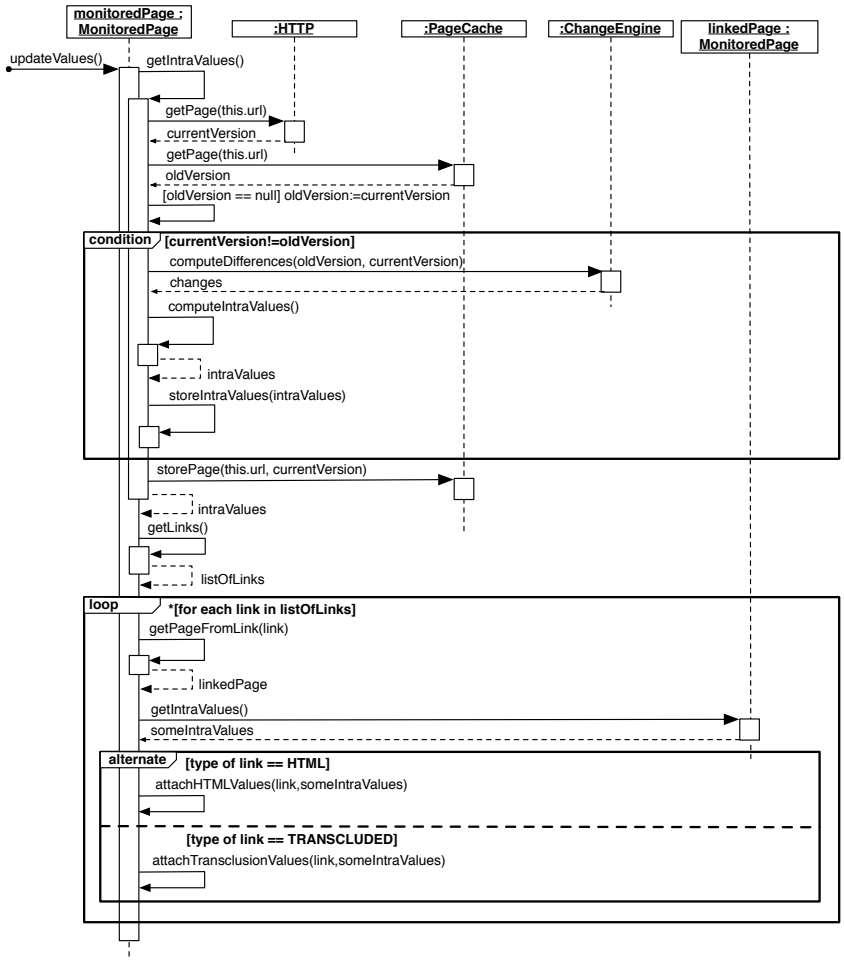

Fig. 2. Awareness mechanism for computing intra– and inter–document awareness.

To compute the awareness information for the inter–document changes, the linked pages need to be accessed. Therefore, the list *listOfLinks* with all links of the *monitoredPage* is retrieved. Note that both *html* and *transclusion links* are included in the list and *listOfLinks* is accessed after the current version of the *monitoredPage* has been stored locally. Therefore the list contains the links included in the current version of the page. Any link that was present in the old version but removed in the current version is not included. Similarly, any link not present in the old version but added in the current version is included. We believe it is reasonable to search for changes only in pages currently linked to the *monitoredPage* as the removal of a link is already depicted as a change in the *monitoredPage*.

In the loop that follows, we handle each of the links separately. We initially retrieve the *linkedPage* and compute the intra–document awareness for it by calling the method *getIntraValues* as for the *monitoredPage*. As explained before, any new version of the *linkedPage* is detected and the changes made to it as well as the corresponding awareness information are computed. Finally, the object *someIntraValues* is returned, including the computed awareness information. *someIntraValues* may be null if the current version of the *linkedPage* is identical to the old version and different than null if the versions differ. Note that if the link was added to the current version of *monitoredPage*, the *linkedPage* is inserted in the *pageCache*. The old version of the *linkedPage* is identical to the current one, so the *someIntraValues* will be null. Changes made to the new linked document will be monitored and reported starting from the moment the link is added. This part of the loop is executed for both *html* and *transclusion links*. The intra–awareness information computed for the *linkedPage*s is then attached to the structure of the *monitoredPage* based on the type of the link. The procedures *attachHTMLValues* and *attachTransclusionValues* are described in Section 3.4.

It is obvious from the procedure described here that we compute inter–document awareness information based on first level links only, i.e. we consider changes made only to pages directly linked to the current page. Although the procedure can easily be adapted for other levels, i.e. links included in linked pages, we believe that the information delivered in this case would not be of additional value to the collaborating situations we address in our current work.

3.3 Intra-document awareness

Since each change made to a page is mapped to an operation, we define the concept of an *opValue* as the effect of the change on the page. The effect of an operation is not a single value, but a collection of values, each of which is computed according to some criteria called *metrics*. Each *metric* addresses specific information and is used to compute awareness information of interest to the users. Therefore, we define the metric $opValue$ as follows.

Definition 1. *$opValue$ is the number of the units of level ulevel that are inserted or deleted by the operation op.*

$opValue(op, ulevel) =$

$$\begin{cases} 0 & \text{if } level(op) > ulevel \\ 1 & \text{if } level(op) = ulevel \\ count(unit_j)_{\substack{unit_j \in content(op) \,\&\, \\ level(unit_j) = ulevel}} & \text{if } level(op) < ulevel \end{cases}$$

Two of the metrics we have defined are the number of characters (*ulevel* = "character level") and the number of sentences (*ulevel* = "sentence level") inserted or deleted by an operation. An operation inserting a sentence of 30 characters would have an *opValue* of 30 if the metric selected by the user is the number of characters or a value of 1 if the metric selected is the number of

sentences. The second metric would be selected if a user wanted to be informed only about "important" changes on the level of whole sentences inserted or deleted. As a result, any operation that inserts/deletes a unit of a lower level than sentence would have an *opValue* of 0 and be omitted from the computation of awareness information. On the contrary, if a user needs to be informed about spelling mistakes as well, then the first metric is chosen. Note that operations that delete content are handled in the same way as operations that insert content with *opValue* computed in the same way and always having a positive value.

The concept of a *nodeValue* is introduced to describe the effect of all changes of a specific type made to a node in a document's structure, i.e. a document part and all the document parts of lower syntactic document levels that belong to it.

Definition 2. *We define the nodeValue of a node N to be the sum of two components: the sum of the opValue of all the operations applied to the node and the sum of the nodeValue of all the node's children.*

$nodeValue(N, ulevel) =$

$$\sum_{op_i \in history(N)} opValue(op_i, ulevel) + \sum_{N_j \in children(N)} nodeValue(N_j, ulevel)$$

nodeValue of a node at the character level is equal to zero, since its history and children are empty sets. Since there can be more than one type of operation, there is also more than one *nodeValue* defined for each node. For instance, a *nodeDeletionsValue* is defined for a node to represent the sum of the *opValue* of all the delete operations applied to the node and the nodes of lower syntactic document levels that belong to it. Further, since there is more than one *metric* to compute an *opValue*, there is also more than one instance of *nodeDeletionsValue*.

A clear advantage of a structured page is the fact that the parts of the page addressed can be of different levels. For example, this means we can compute the total changes made to each of the paragraphs included in the *Group Main Page* shown in Figure 1, or for each of the sections, etc. The value computed for a specific section reflects the number of paragraphs inserted or deleted to the section and the number of sentences inserted or deleted to the paragraphs etc.

A more detailed description of our awareness mechanism presented in this section can be found at [13]. The concepts described above are applicable to collaborative authoring of structured documents of any type. We already used them to compute awareness information on co–authored text and graphical documents. The formal definition of the concepts as well as details of their use for graphical and text documents can be found in [10] and [13], respectively.

3.4 Inter-document awareness

We now describe the procedure followed to attach to a *monitoredPage* the awareness information computed for a *linkedPage*. As described in Section 3.2, the

awareness information is computed in the same way for all *linkedPage*s linked through either an *html* or a *transclusion link* to a *monitoredPage*. This awareness information comprises a set of *nodeValue*s computed for each of the parts of the *monitoredPage*. To keep the awareness information delivered through the *monitoredPage* up–to–date, we need to attach the computed values to the parts of the *monitoredPage* that are affected.

If the link to the *linkedPage* is an *html link*, the part of the *monitoredPage* where the link is included needs to be updated. We decided to attach to this part all the awareness information computed for the *linkedPage*. In this way, we provide the user the possibility while browsing through the *monitoredPage* and watching intra–document changes, to also be informed about the changes made to the *linkedPage* without loading the page. Our aim is to deliver to the user an edit profile with the quantity of the changes in the *linkedPage*, as well as the parts of the page that changed. In this way, the user can decide whether they need to revisit the linked page and update the information included in the *monitoredPage*. The way the awareness information is presented to the user is shown in Section 4.

If the link to the *linkedPage* is a *transclusion link* then only part of the awareness information computed for the *linkedPage* needs to be transferred to the *monitoredPage*. Through the transclusion mechanism, the part of the *monitoredPage* that is transcluded from a *linkedPage* is updated to include the latest changes made to it. However, updating the transcluded part in the *monitoredPage* is not enough. Awareness information that is attached to this part, needs to be transferred to the *monitoredPage* as well so that users are informed about the amount of changes. To do so, the awareness information computed for the *linkedPage* is scanned and only the values computed for the transcluded part are kept. This mechanism is heavily dependent on the transclusion mechanism and the metadata that is created during the authoring process of the *transclusion link*. This metadata is usually new tags inserted in the transcluded or the composed pages, or other anchors that can be used to define the beginning and end of the transcluded part [14]. Using this data, we identify the transcluded part in the *linkedPage*, extract the awareness information computed for it and copy it to the corresponding part of the *monitoredPage*.

4 Visualisation of intra/inter–document awareness

We revisit our motivating example to present the tools used to visualise the computed awareness information. We use only a representative sample of changes made to two of the example pages to demonstrate how the visualisation tool informs users about different kinds of changes. We assume there are some intra–document changes (annotated with "A" in Figures 3 and 4) made to the main page and the internal page of project 1 (interactive paper project). The main page has an *html link* to the project page through the phrase "interactive paper" in the first paragraph (B) and the paragraph about the interactive paper project is a transcluded paragraph (C) from the project page. The changes made to

the main page are insertions of text in the paragraph about group research. The changes made on the project page are an insertion of a word in the first paragraph (the transcluded paragraph), the insertion of two new paragraphs (the sixth and seventh) and the insertion of a new member in the list of group members (paragraph number nine). Note that titles are handled as paragraphs of one sentence and changes of the project page are considered as inter–document changes for the main page. Finally note that an external page linked through the main page could also be used instead of the internal project page and the computed awareness information would be visualised in the same way.

We assume that the main page contains 10 paragraphs. The window in Figure 3 presents the first 6 paragraphs of them.

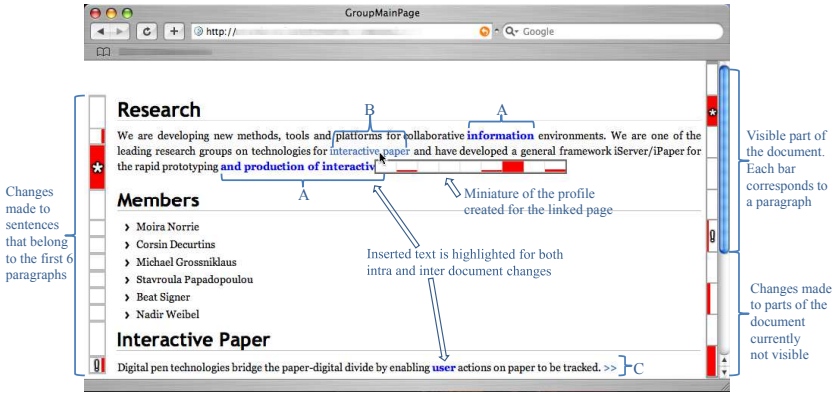

Fig. 3. Visualisation of intra– and inter–document changes made to the main page.

In Figure 3, we also show how we visualise the intra–document changes of the main page. Two edit profiles are introduced on the sides of the page. The profile on the right side shows an overview of all changes made throughout the page, while the profile on the left shows all changes made to the portion of the page currently shown. In order to construct the profile on the right side presented in Figure 3, the awareness information computed for each paragraph is used. By using the profile, the user can instantly spot that there have been a lot of changes on the second and the last paragraph and some changes in the sixth and eighth paragraph. The left profile can be thought of as a zoomed version of the right profile. The information provided by the left profile corresponds to the first six paragraphs, which is the part of the right profile marked by the scrollbar.

The reason for the existence of the left profile, is that the information provided by it can be on a different syntactic level. In Figure 3, for instance, the left profile shows changes made to each of the sentences of the first six paragraphs. In this way, a user that would notice the large amount of changes made to the second paragraph, could further filter the awareness information and find how the changes in the paragraph are distributed to the paragraph sentences. The

height of the bar corresponding to each sentence is adjusted to the length of the sentences for the left profile, to ease the mapping of changes to document parts. The width of a bar in each profile represents the normalised number of words inserted or deleted in the corresponding document part. Please note that a user is able to configure the syntactic levels used in the profiles as well as the detail of the information provided through them to visualise information about changes of various types on a user-defined granularity. The intra–document changes made to the project page are also visualised in the same way as shown in Figure 4.

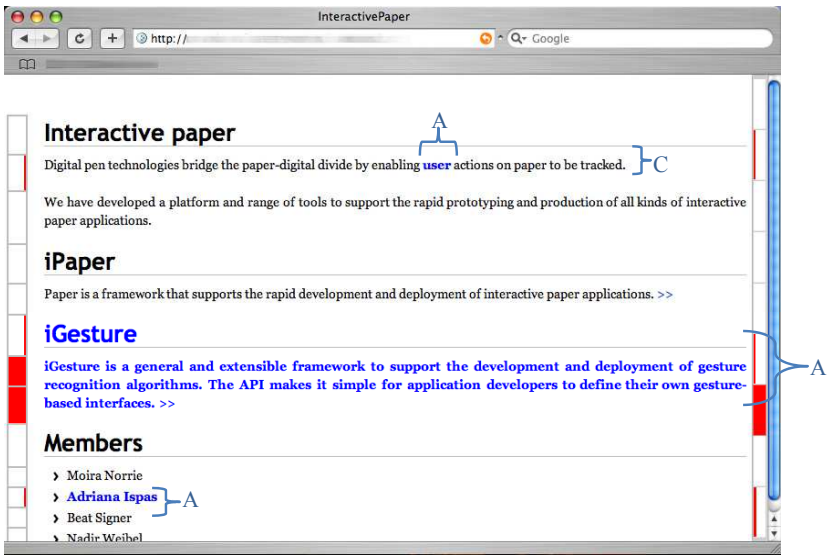

Fig. 4. Visualisation of intra–document changes made to the project page.

Since the first paragraph of the main page includes an *html link* to the project page, changes made to the project page might influence the content of the main page. For instance, the introduction of a new prototype (iGesture) in the project page might need to be included in group research summary (first paragraph) in the main page. Hence, when a user points with the mouse on the html link, an overview pops up that shows the changes made throughout the project page. We expect this feature to give the user a rough idea of the amount of changes made to the project page as well as the parts of the page that changed the most to help the user decide whether they need to visit the project page to see the changes in detail and whether the content of the main page should be modified as well. With the "*" symbol on the profiles we inform users if there are changes made to pages linked through *html links*.

The changes made to transcluded parts are presented as intra–document changes through the edit profiles. The sixth paragraph of the main page is a transcluded paragraph. The changes made to it are presented through the two

profiles in the main page, exactly in the same way they are presented in the project page. The distinction between the transcluded and compound page is made through the "!" symbol added to the compound page at the parts of the profiles that correspond to the transcluded parts.

Our awareness mechanism and the visualisation tool introduced above aim to increase the amount of awareness information presented to users about changes made to a page and the pages linked to it. In the absence of adequate awareness information, a number of problems could appear. For instance, a user editing the main page would not be informed about changes made at the linked page (interactive paper project). They would not know that the list of members in the interactive paper project is updated since a new member was added. As a result, the list of members in the main page would not be updated to include the new member. Additionally, users would not be informed about the new prototype that is released in the interactive project and therefore, information about the released frameworks on the main page would remain obsolete. Our awareness mechanism supports the users to avoid all of the above situations.

5 Related work

A closer look at CSCW research reveals various approaches developed to provide change awareness in text-based collaborative applications [2, 3], software development collaborative tools [4] and applications with 2D graphical scenes of objects [1]. However, not much work has been done on providing awareness of changes in the co-authoring of web pages.

In [15], a notification mechanism is proposed for tracking changes made to web pages. Users can mark regions of interest on a web page that are stored as bitmaps. The system periodically checks if the selected regions visually differ from the stored ones with a certain severity index fixed by the user. If an interesting region is modified, the user is notified by the system. This approach is image-based and assumes the web page retains its spatial layout. In contrast, our approach provides an awareness mechanism that is attached to the document structure and therefore is independent of page layout changes. Moreover, our approach quantifies changes.

Tools for computing the difference between HTML web pages [16] present superimposed on the document, the changes made to it. While this enables the users to detail the changes made by their collaborators, it does not provide any support in finding "hot areas" of the document where "important" changes have been made. The lack of an overview of all changes made throughout the web page, forces the users to scroll through the document to find areas with "interesting" changes. This can be frustrating and time inefficient especially in case of long documents. A second drawback of existing approaches is that they cannot measure the effect of a change made to a document, or cannot distinguish changes on different syntactic levels. For instance the fact that an insertion of a two–word sentence has a different effect from an insertion of a twenty–word sentence is not depicted by existing collaborative applications. Both

are presented as one change. Finally, it is not possible for a user to be informed separately about spelling mistakes, i.e. changes that altered only some characters and separately about insertions and deletions of whole paragraphs.

Notification tools, such as Watch That Page [17], ChangeDetect [18] and Website Watcher [19], track changes made to specified web pages. Some of them, for instance Website Watcher [19], allow the possibility to automatically add to the list of monitored pages those pages linked to the main web page. Users are then notified if changes are made to one of the monitored pages. However, no information is provided about the relation between the monitored pages and how changes in a linked web page might influence the changes in the main web page. Our approach keeps users aware about changes made to pages linked to the current web page and, in this way, helps users avoid semantic inconsistencies.

Wikipedia [20] provides users with the possibility of tracking changes made to a wiki page by means of a revision history attached to the page. It consists of the old versions and a record of the date and time of every edit, as well as the user who made it. Wikipedia also offers the "watch pages" feature by which users receive email notifications when specified pages change. The notification mechanism provides the number of added or deleted bytes to each page. With our approach, in addition to the intra-document awareness, we provide inter-document awareness where we track changes on linked documents to the web page that might influence the content of that web page. We track changes even on the documents identified by the links added during the process of collaboratively authoring the web page. Moreover, by means of the edit profile, we present changes at different granularity levels.

In [21], an overview of the evolution of a wiki page is presented using a history flow visualisation to provide information about how a group contributed to the web page or how a modification influenced the current version of the web page. Unfortunately, no awareness mechanism is offered for the changes made to the linked pages to the current wiki page. Moreover, the document evolution is computed only on the document level and no information is provided on the changes made to parts of the document.

6 Conclusion and future work

We have presented a change awareness mechanism that tracks intra- and inter-document edits in the co-authoring of web pages. We described how metrics are computed to quantify the changes made inside a document as well as on transcluded parts of the document and linked documents. Moreover, we described a visualisation tool based on edit profiles that enable users to have an overview of changes done on a web page and any pages linked to it. This allows users to easily spot "interesting" changes done on document parts and browse these changes on finer granularity levels.

We plan to test the usability of our approach by conducting user studies on the prototype that implements the ideas described in this paper. During user studies we plan to experiment with various visualisations. Finally, one of our

future work directions is to extend our awareness mechanism to take into account not only physically linked documents, but also semantically linked documents in the context of semantic wikis or more general semantic web.

References

1. Tam, J.R.: Supporting change awareness in visual workspaces. Master's thesis, Department of Computer Science, University of Calgary, Alberta (2002)
2. Hill, W.C., Hollan, J.D., Wroblewski, D., McCandless, T.: Edit wear and read wear. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 1992), Monterey, CA, USA (1992) 3–9
3. Neuwirth, C.M., Chandhok, R., Kaufer, D.S., Erion, P., Morris, J., Miller, D.: Flexible diff-ing in a collaborative writing system. In: Proceedings of the ACM Conference on Computer-supported Cooperative Work (CSCW 1992), Toronto, ON, Canada (1992) 147–154
4. Eick, S.G., Steffen, J.L., Eric E. Sumner, J.: Seesoft-A Tool for Visualizing Line Oriented Software Statistics. IEEE Transactions on Software Engineering **18**(11) (1992) 957–968
5. Nelson, T.: Literary Machines. Mindful Press (1982)
6. Krottmaier, H., Helic, D.: Issues of transclusions. In: Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (E-Learn 2002), Montreal, Canada, AACE (2002) 1730–1733
7. Papadopoulou, S., Ignat, C.L., Oster, G., Norrie, M.C.: Increasing awareness in collaborative authoring through edit profiling. In: Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2006), Atlanta, GA, USA (2006)
8. Papadopoulou, S., Reuss, E., Norrie, M.C.: A user study of edit profiles in collaborative authoring systems. In: Proceedings of the International Symposium on Collaborative Technologies and Systems (CTS 2008), Irvine, CA, USA (2008)
9. Papadopoulou, S., Norrie, M.C.: Shadow document sets for synchronously-aware asynchronous collaboration. In: Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007), New York, NY, USA, IEEE Computer Society (2007)
10. Papadopoulou, S., Ignat, C.L., Norrie, M.C.: Awareness model to overview modifications in collaborative graphical authoring tools. In: Ninth International Workshop on Collaborative Editing Systems (IWCES 2007) - The 2007 International ACM Conference on Supporting Group Work (GROUP 2007), Sanibel Island, FL, USA (2007)
11. Ignat, C.L., Norrie, M.C.: Customizable Collaborative Editor Relying on treeOPT Algorithm. In: Proceedings of the European Conference on Computer-supported Cooperative Work (ECSCW 2003), Helsinki, Finland, Kluwer Academic Publishers (2003) 315–334
12. Myers, E.W.: An $O(ND)$ difference algorithm and its variations. Algorithmica **1** (1986) 251–266
13. Papadopoulou, S., Norrie, M.C.: How a structured document model can support awareness in collaborative authoring. In: Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007), New York, NY, USA, IEEE Computer Society (2007)

14. Kolbitsch, J., Maurer, H.: Transclusions in an html-based environment. Journal of Computing and Information Technology **14** (2006) 161–174
15. Greenberg, S., Boyle, M.: Generating custom notification histories by tracking visual differences between web page visits. In: Proceedings of Graphics Interface (GI 2006), Quebec, Canada (2006) 227–234
16. W3C HTML Diff service (2008) `http://www.w3.org/2007/10/htmldiff`.
17. WatchThatPage. Your monitor for changes on the web (2008) `http://www.watchthatpage.com/`.
18. ChangeDetect. Be the first to know (2008) `http://www.changedetect.com/`.
19. Website-Watcher. Save time, stay informed (2008) `http://aignes.com/`.
20. Wikipedia, the free encyclopedia that anyone can edit (2008) `http://www.wikipedia.org/`.
21. Viegas, F.B., Wattenberg, M., Kushal, D.: Studying cooperation and conflict between authors with history flow visualizations. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2004), Vienna, Austria (2004) 575–582