Providing Awareness in Multi-synchronous Collaboration Without Compromising Privacy

Claudia-Lavinia Ignat
INRIA Nancy Grand-Est
LORIA, France
ignatcla@loria.fr

Stavroula Papadopoulou
Department of Computer Science
ETH Zurich, Switzerland
papadopoulou@inf.ethz.ch

Gérald Oster
Nancy-Université
LORIA, France
oster@loria.fr

Moira C. Norrie
Department of Computer Science
ETH Zurich, Switzerland
norrie@inf.ethz.ch

ABSTRACT
When involved in collaborative tasks, users often choose to use multi-synchronous applications in order to concurrently work in isolation. Hence, privacy of their changes is maintained until they decide to publish their contributions. Not being aware of changes made by their collaborators, they often create concurrent modifications which might generate conflicts or lead to redundant work. We propose an awareness mechanism that solves this problem by computing and providing awareness in multi-synchronous collaboration while at the same time respecting user privacy by allowing users to specify the detail of information made available to their collaborators. The computation of awareness is based on metrics that measure the effect of changes for the different types of changes, on the different syntactic document levels and document parts. For the visualisation of awareness, we employ the concept of edit profiles.

ACM Classification Keywords
C.2.4 Computer-Communication Networks: Distributed Systems—*Distributed applications*; D.2.2 Software Engineering: Design Tools and Techniques—*User interfaces*; H.5.3 Information Interfaces and Presentation: Group and Organization Interfaces—*Asynchronous interaction, Synchronous interaction, Computer-supported cooperative work.*

General Terms
Human Factors, Design, Experimentation.

Author Keywords
CSCW, Awareness, Privacy, Edit profiles, Multi-synchronous interaction.

INTRODUCTION

Many tasks in business and academia involve groups of individuals working together to achieve some common goal. Not surprisingly, the development of collaborative authoring tools therefore became a central topic of interest in the CSCW (Computer Supported Cooperative Work) community. According to the synchronicity of writing activities, collaborative authoring tools can be classified as either synchronous, asynchronous or multi-synchronous. Synchronous authoring tools, also referred to as real-time collaborative authoring systems, imply that changes made by one user are immediately transmitted to other group members. Asynchronous authoring tools support groups of individuals who contribute at different times by taking turns of revising and editing the shared artifact. Multi-synchronous authoring tools [7] allow simultaneous work in isolation of members of the group and the subsequent integration of their contributions.

An example of asynchronous collaboration is the traditional collaboration over email. When people want to collaboratively author a document, they send each other emails containing as attachment versions of the document that integrate their changes. This work mode requires a good planning of activities [17]. Integration raises no difficulties if people work sequentially. Manual integration in the case of parallel work can be easily performed if the document is well segmented and different document segments are assigned to different authors. However, if the document decomposition is not possible, manual integration of parallel changes becomes complex. Recently there has been increasing interest in real-time editors such as GoogleDocs [1], SubEthaEdit [2] or synchronous pair programming [14] for supporting parallel work. However, multi-synchronous authoring tools continue to be largely used since they offer support for private work while working in parallel. Private workspaces are essential for collaboration [18] as they give co-authors the possibility to carry out polishing and revision of their contributions before communicating or including them in shared documents. Multi-synchronous collaboration is also widely used for collaborative software development where work in isolation ensures that the source code is always kept in a consistent state and can be compiled. Therefore, users often work with version control systems, such as CVS[3] or Subversion [5], for the development and documentation of software projects, or with wiki systems, such as Wikipedia, for the collaborative writing of documents over the web.

Awareness, defined by Dourish [8] as an "understanding of the activities of others which provides a context for your own activity", has been identified by the CSCW community as

CSCW'08, November 8–12, 2008, San Diego, California, USA.

one of the most important issues in collaborative document authoring, independently of the collaboration mode. A special type of awareness, *change awareness*, was defined as "the ability of a person to track the changes that other collaborators have made to a group project" [25].

Extended research has been conducted by various groups to produce collaborative applications that provide change awareness in synchronous, asynchronous or multi-synchronous collaboration. Unfortunately, the awareness information provided by the currently available awareness–enabled multi-synchronous applications is restricted, mainly due to the nature of the collaboration. When users work with multi-synchronous collaborative applications, they remain isolated. They publish their changes to other users when they commit their changes to the repository and are informed about changes of other users only when they update their local copy. Some of the existing change awareness approaches [22, 24] inform users about changes made to a document by highlighting the changes made by other participants over time.

These approaches offer awareness for the changes committed in the repository and integrated in the local workspace of the user. However, when users are not aware of other users' activities while they are working, they may concurrently modify the same parts of the shared document. These are called blind modifications [13] and might occur due to concurrent changes that are either uncommitted, or committed but not yet integrated by the user. This could lead to conflicts or redundant work. A conflict would, for instance, be generated if a user proofreads a document section while another user concurrently deletes this section. Finally, redundant work would occur if two users concurrently perform identical tasks on their local copies of a document.

Notifying users when changes are committed is a form of awareness that has been adopted by some systems [10]. Other approaches [6, 20] send changes in real-time to all users to provide real–time awareness information even in multi-synchronous collaboration. Sending operations immediately after their execution to other users would eliminate the issues of blind modifications, but they violate user needs of working in isolation and therefore compromise user privacy. In [13], the notion of *ghost operations* was proposed to deal with the tradeoff between awareness and privacy. The main idea of this approach is that local uncommitted operations can be filtered according to user preferences before being sent to other users. An operation is filtered according to various filters depending on the privacy relation of the local user with the remote one. In this approach, rather than integrating the received ghost operations on the document state and computing awareness information, these operations are used only to annotate the document.

The goal of our work is to create an awareness mechanism that can be applied to multi-synchronous collaborative applications to

- provide real–time awareness information and hence prevent the creation of blind modifications,
- respect users' privacy by allowing users to define the amount of information to be transmitted to their collaborators,
- compute awareness information on various levels of granularity based on user preferences, and present the changes made along a document using flexible visualisation tools,
- provide all the above information for both committed and uncommitted changes.

To successfully provide the above information in a multi-synchronous environment, we use ghost operations which are sent to all collaborators in real-time, i.e. right after their generation. We also use edit profiles [19] for a flexible computation and visualisation of awareness information. Edit profiles enable users to have a quick overview of the past changes and of concurrent changes made by other users. Concurrent changes include changes that are either committed or uncommitted and released to the other users by means of ghost operations defined according to user privacy preferences. Finally, to provide awareness information on different granularity levels, we choose to use a structured document model, for instance a hierarchical model. In this way, we are able to address specific document parts of various syntactic document levels, computing and presenting awareness information about them. Hierarchical models encompass a large class of documents such as textual and XML documents which make our approach applicable to a large number of existing multi-synchronous applications.

The main contribution of this paper is an awareness mechanism where users filter the amount of information about their changes to be delivered to their collaborators, based on user defined privacy levels, and visualise the filtered updates. Our approach makes users aware of concurrent changes so that they can initiate communication with the users that made those changes. In this way, conflicts can be avoided or resolved at an early stage.

Our paper is structured as follows. We first review existing awareness approaches in multi-synchronous collaboration. We then present the document model that we adopted and the representation of operations that we used for communication in multi-synchronous collaboration. We go on to present an awareness mechanism based on edit profiles for multi-synchronous communication. Further, we show how we extended this awareness approach to address privacy issues. We provide a general architecture for our approach and introduce the notion of ghost operations along with various filters that can be used for their generation. By means of an example, we present an edit profile for the visualisation of ghost operations. We also show how awareness information is computed from data provided by ghost operations. Finally, in the last section, we provide concluding remarks and some directions for future work.

RELATED WORK

Most awareness approaches for multi-synchronous communication focussed mainly on change awareness. These approaches highlight changes made by other participants to an

artifact such as a document or workspace since the last time the user saw that artifact. An initial framework on change awareness was proposed in [11] and then refined in [24]. These approaches keep a user aware about changes that were made and published while they were working in isolation. They do not present changes that are concurrently made and not yet published and therefore, these approaches do not prevent blind modifications.

The State Treemap [15] is an awareness widget designed to inform users about states of shared documents. Different states are defined for a document such as LOCALLYMODIFIED, POTENTIALLYCONFLICT – when two copies of the document are modified and none of the changes are published yet – or WILLCONFLICT – when a document copy is modified locally and some changes on that document have been committed. However, the granularity of the awareness information is the document and therefore it is impossible to measure the divergence between two copies in terms of concurrent modifications within the document.

Palantìr [23] provides awareness information about concurrent modifications done in isolation in the context of configuration management systems. It is based on the same principle as State Treemap, the main difference being that severity information that computes the amount of changes made among documents is added. Unfortunately, the granularity of provided information is still the document. Moreover, the severity metrics do not provide enough information to infer changes that could cause potential conflicts at the merging phase.

Concerning quantitative measurement of divergence between document copies, the approach proposed in [16] provides divergence metrics. Contrary to Palantìr and State Treemap approaches, metrics are computed based on operations modeling concurrent changes and not on events triggered by document state transitions. Merging of concurrent operations is simulated in real-time on each site, making it possible to compute various metrics. For instance, it is possible to compute the amount of changes made on each document as in Palantìr, but also the amount of conflicting/overlapping changes. However, this approach does not deal with issues of privacy and all local changes are sent in real-time to the other users.

In [20], the authors proposed an edit profile that counts the number of operations generated by users on different parts of a document. The operations taken into account are both committed and uncommitted. However, as in [16], all generated operations are transmitted immediately to the other users and no privacy issues are considered.

Another approach for providing awareness information in real-time while working multi-synchronously was proposed in [6]. The approach is adapted for collaborative software development and provides developers with warning messages concerning concurrent activity and the possibility to consult a list of conflicts. Based on a selected conflict, a user can set watches for concurrently edited elements. For instance, they can be notified when a collaborator has finished editing the element. However, no quantitative measure is provided for concurrent operations or for conflicts. Moreover, all uncommitted operations are sent to all members of the team and no privacy issues are taken into consideration.

The approach described in [13] deals with privacy issues in multi-synchronous communication by means of ghost operations that represent filtered operations according to the privacy settings of a user with respect to other users. Although the operations are used to annotate the document parts with concurrent changes, no quantitative measures are provided to compute the divergence between two copies of the document.

A "cloudburst model" has been proposed in GROVE [9] synchronous multi-user editor to hide changes recently made by remote users. The position and size of a cloud indicates the approximate location and extent of the modification. After certain periods of time, the clouds disappear and are replaced by the actual content of the modifications. However, this filtering mechanism was applied for reducing distraction rather than for preserving privacy and is applied at the receiver side rather than the sender side.

DOCUMENT MODEL AND OPERATIONS

In this section, we present the model of the document that we adopted and the classification and representation of operations for communication in multi-synchronous collaboration.

We adopted a hierarchical structure of the document as it encompasses a large class of documents such as textual and XML documents. For instance, a book contains chapters composed of sections. Each section is composed of paragraphs, each paragraph of sentences, each sentence of words and each word of characters. In this case, the granularity levels associated with the hierarchical model would be book, chapter, section, paragraph, sentence, word and character. We represent the node of a document as described in [12].

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

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

In Figure 1, an example of a document with the 5 levels of granularity - document (level 0), paragraph (level 1), sentence (level 2), word (level 3) and character (level 4) - is illustrated. A history is assigned to each node, containing the operations referring to children of that node. For example,

the log of operations associated with a paragraph includes insertions and deletions of sentences in that paragraph.

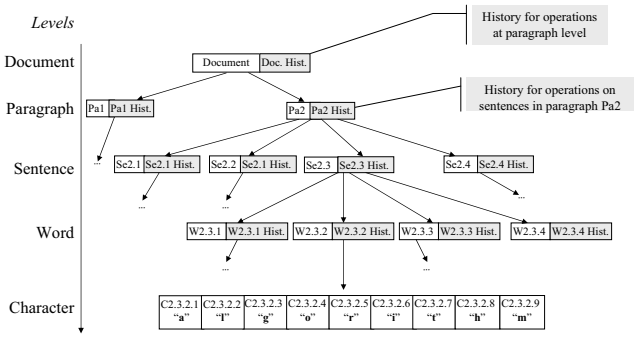

Figure 1. Hierarchical representation of a text document.

Operations representing changes made on the hierarchical structure have been defined in a similar way to that described in [12].

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.

The vector position specifies the indexes that compose the path in the tree where the operation is applied. For instance, if an insert operation of word level is applied to the text document of our example, information about the paragraph and the sentence in which the word is located, as well as the position of the word inside the sentence, are included in the position parameter.

Finally, the insertion of the sentence "CSCW stands for Computer Supported Cooperative Work." would have a length vector equal to $< 0, 1, 7, 52 >$ since it inserts 0 units of paragraph level or 1 unit of sentence level or 7 units of word level or 52 units of character level.

Usually multi-synchronous collaborative systems such as version control systems maintain a central repository containing versions of the document. It offers an optimistic approach to version control by allowing users to work on the same copy. Users can check out a copy of the current document, make changes on the copy of the document and commit the copy back into the repository. In our work, we assume this kind of collaboration. We make a distinction between operations that are *committed* and *uncommitted*. While working on their local workspace, users should be made aware about committed and uncommitted operations generated in parallel with the local changes.

AWARENESS FOR MULTI-SYNCHRONOUS COLLABORATION

Taking advantage of the underlying structure of a document, as well as the use of operations to represent changes made to the document, we created a flexible awareness mechanism that computes and visualises change awareness information for the different document levels and document parts. We briefly present here the main concepts of this procedure to ease the description, in the next section, of the extension of this mechanism to compute awareness information in the light of privacy issues and ghost operations.

The use of a structured document model allows us to address each document node separately, independently of its level. Tracking all changes made to a document, i.e. all the operations created in the document, we can find the changes made to each document node and its children nodes, compute the effect of these operations and present the results through proper visualisation tools for different syntactic document levels or document parts. For instance, accessing the computed information for all the paragraphs in a document and presenting them via an edit profile [19] shows to the user the amount of changes made to all the paragraphs in a document. Accessing the information computed for all the paragraphs of a specific section presents a zoomed version of the previous example, by presenting awareness information about the specific document part. Summarising, our awareness mechanism computes and visualises awareness information at the different document levels and for the different document parts.

When an operation is received at a site, the document node is found where the operation needs to be applied and the operation's effect to the node and its ancestors is computed and stored. The computation of the operation's effect produces a value, called *opValue*. Since the effect of an operation may vary depending on user needs and some application specific characteristics, we have created various metrics, and use them to compute various instances of an operation's *opValue*. The text collaborative application of our example has a structured document model and syntactic document levels from character to document level. The metrics defined for it could be the character, word, sentence or paragraph level, representing the operation's effect at the node where it is applied by means of the number of characters, the number of words, the number of sentences and the number of paragraphs inserted/deleted by the operation to/from the document node. An insertion of a sentence at a paragraph node has an *opvalue* equal to 1 if the selected metric is the sentence level, an *opValue* equal to the number of the words inserted by the sentence if the metric is the word level etc. In this way, our awareness mechanism computes awareness information at a granularity specified by user needs, roles and preferences. For instance, an non-native english

speaker who receives a document corrected from a proofreader would probably like to visualise spelling mistakes, i.e. changes of characters in word level nodes. On the contrary, an editor in a publishing company who needs to have an overview of the authoring progress and current state of a document, would probably only need to be informed about major changes made to the document such as insertions and deletions of entire paragraphs.

For each document node, the above computed *opValues* are grouped based on the user who created the operation, the operation type and the metric used to compute the value. The *opValues* of each group are summed to create the corresponding *nodeValue*. This procedure results in multiple instances of *nodeValue*s as well. In synchronous collaboration, or multi-synchronous collaboration where no privacy issues are considered, the remote operations that arrive at a site have all the required information needed to compute the *opValue*s and *nodeValue*s. After the computation of all the above values, the ones corresponding to the document level, document part, operation type etc. chosen by the user, are selected and visualised through an edit profile.

We conducted user studies investigating the usefulness of edit profiles as a mechanism to deliver real-time awareness in multi-synchronous collaborative environments. In [21] we reported on user feedback concerning the use of edit profiles as a tool that delivers awareness information. However, users also expressed their concerns considering the invasion of their privacy when involved in such collaboration situations. We therefore propose, in the next section, an extension of our awareness approach to deal with privacy issues.

AWARENESS IN THE LIGHT OF PRIVACY ISSUES

The awareness mechanism described in the previous section is general enough to be applied to any kind of collaborative application with a structured document model, independently of the document type or the mode of collaboration. However, it cannot be applied to privacy–sensitive collaborative situations because it assumes that all the required information for the computation of awareness is sent by users to their collaborators. However, very often this is not the case, since one of the main reasons for users working in a multi-synchronous environment is that they want to work in privacy and review their work before publishing it.

Which data is considered private cannot be uniquely defined. It depends on users, their collaborators, their roles, the collaborative situation and the task to be accomplished. Therefore, user privacy level should be set by users themselves depending on their current needs and situation. The computation of awareness information in privacy–sensitive environments should respect any private uncommitted data and use the available information only. Our work concentrates on extending the awareness mechanism previously described to compute the *maximum available* information in the light of ghost operations.

In what follows, we describe an overview of the general procedure for the creation and receipt of ghost operations and the computation of the awareness information provided by ghost operations. To analyse in detail the generation of ghost operations, we first give their definition by means of the possible filters applied to the attributes of a real operation. Then, we present some of the masks that can be used combining the various filters and discuss example situations where each mask could be used. Finally, by means of edit profiles, we show how the information concerning the ghost operations is presented to the users in the case of an example situation and discuss in detail the computation of the values presented in the visualisation tool.

Architecture

The architecture shown in Figure 2 presents the procedure followed from the creation of an operation at a user's local site to the receipt of it by another user and the computation of awareness information. The steps followed are:

- Generation of a real operation at a user's site.
- Filtering of the operation attributes by using various masks to create a ghost operation.
- Transmission of the operation through the network to all collaborators.
- Receipt of an operation from a collaborator and extraction of the available information. The document node where the operation will be applied is found, the appropriate *opValues* are computed and the corresponding *nodeValues* are updated.

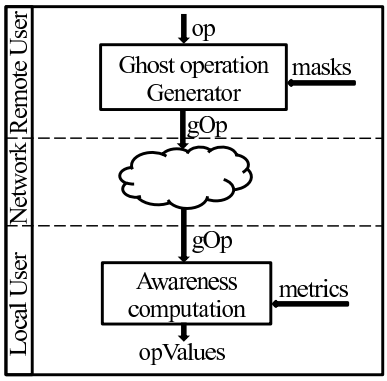

Figure 2. Generation of ghost operations, transmission through the network and computation of awareness after the operation's receipt.

In what follows we analyse in more detail the second and fourth item of the above procedure, i.e. the *ghost operation generator* and the *awareness computation*.

Ghost operations

In [13], a *ghost operation* was defined as $g(operation) = < filter(type), (filter(parameter))* >$ following the definition of the original operation given by a type and a list of parameters *operation=<type,(parameter)*>*. Hence, the ghost operation is the operation obtained by filtering the type and the parameters of the original operation according to user privacy preferences. The inspiration for the filtering

mechanisms applied to operation parameters was the blurring filtering mechanisms applied to video documents for masking several details while still providing overview usefulness for awareness [4].

Ghost operations for the document of our example where op =<*type*, *level*, *position*, *content*, *length*, *user*> are defined as $g(op)$ =<*filter(type)*, *filter(level)*, *filter(position)*, *filter(content)*, *filter(length)*, *filter(user)*>, where

- filter(type)= $\begin{cases} insert, & \text{if } op \text{ is an insert and its type is not masked} \\ delete, & \text{if } op \text{ is a delete and its type is not masked} \\ edit, & \text{if } op \text{ is a delete or insert and its type is masked} \end{cases}$
- filter(level)= $\begin{cases} level, & \text{if } op \text{ has the level unmasked} \\ null, & \text{if } op \text{ has the level masked} \end{cases}$
- filter(position=$[V_0, V_1, \cdots V_n]$)= $[V_0, \cdots V_i]$, where $0 \leq i \leq n$
- filter(content)= $\begin{cases} content, & \text{if the content of } op \text{ is not masked} \\ null, & \text{if the content of } op \text{ is masked} \end{cases}$
- filter(length)= $\begin{cases} length, & \text{if the length of } op \text{ is not masked} \\ null, & \text{if the length of } op \text{ is masked} \end{cases}$
- filter(user)= $\begin{cases} user, & \text{if the user identity of } op \text{ is not masked} \\ null, & \text{if the user identity of } op \text{ is masked} \end{cases}$

The vector of positions of the original operation can be filtered by providing the positions of the higher level of granularity of the node, but hiding the positions of the lowest level of granularity. For instance, for an original operation of insertion of a word, only the position of the paragraph where the word is to be inserted can be provided and the sentence and word level positions can be hidden.

Privacy levels

As discussed above, the level of privacy requested from users may be task or user specific. Therefore, we decided to make the filtering/masking mechanism that creates the ghost operations quite flexible by offering various levels of privacy. Here we present the most important ones in terms of the operation attributes that get masked in each level. We also give examples of collaborative situations where each level could be selected. We realise that the possible masks to be used are equal to the maximum number of combination of the filters presented above and therefore a lot more than the ones presented here. We believe that some of them would not make any sense and possibly not be used at all, while others could be used frequently. Therefore we decided to introduce the ones we expect to be needed most often by users.

No privacy: We start from the first level where no privacy issues arise, and therefore no information is masked. These ghost operations hold the maximum information and the computation of awareness is not restricted at all. All the *opValue*s are computed and the *nodeValue*s of the document nodes affected by the operation are updated.

Mask the user and / or type: Here the *type* or the *user*, or both of these attributes can be filtered to produce the ghost operation. This level of ghost operations will be created in collaborative situations were the users want to keep their anonymity until they commit their changes, or when they want to inform their collaborators that the document is being edited without specifying the type of changes. The exact document part that is modified is available to the collaborators and information about the extent of the changes as well. Users receiving such ghost operations will be notified about the document parts being edited and as a result are likely to avoid working on the same document areas. We believe it would also be reasonable to mask the *content* as well if the type is masked. An example of such a ghost operation is:

gOp1= $< edit, 2, [2, 3, 4], null, [0, 0, 1, 4], null >$.

Independently of the *type* and/or *user* being filtered, the next levels concentrate on the filtering of other attributes.

Mask the changes: This level holds operations where the *content* is filtered. Such ghost operations might be produced in collaborative situations where the users allow the exact document part being edited and the extent of the changes to be available to their collaborators, but do not publish the actual changes. An example of such a ghost operation is:

gOp2= $< insert, 2, [2, 3, 4], null, [0, 0, 1, 4], null >$.

Mask the changes and their effect: Ghost operations that conform to this level of privacy have the *content* and the *length* masked. Users who intend to make changes to a specific paragraph in a document but do not yet know what exactly to write, often produce a lot of changes which they later discard. To mask such phenomena but at the same time inform their collaborators where they are currently active, ghost operations of this level could be used. The remaining unmasked attributes, i.e. the *level* and *position* can be used by the user who receives the ghost operation to check if the position is also filtered and distinguish this case from the next one. For instance, an operation of word level, where the position is not masked, should have a position attribute, where a complete path from the document level to the word level is given. An example of such a ghost operation is:

gOp3= $< edit, 2, [2, 3, 4], null, null, null >$.

Mask the changes, their effect and part of the position: Finally, we consider collaborative situations were the user also wants to keep private information about the exact document part where the changes are made. This can be achieved either by hiding the *position* attribute (part of it or all of it), or by hiding the *level* attribute, or both. Filtering the *level* attribute will make it impossible to verify whether the posi-

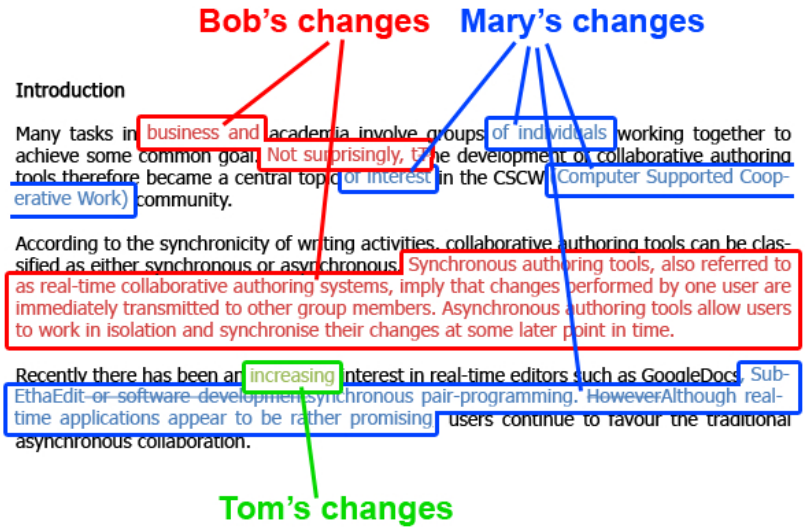

Figure 3. Concurrent changes generated by Mary, Bob and Tom

tion is incomplete and hence the position will also be considered masked. Finally, filtering part of the *position* means making available only part of the path in a structured document where the operation will be applied. For instance, a proofreader might want to inform their collaborators about the document part they are working on, without giving details about the exact sentences or words being edited. Such information could be enough to inform collaborators about their activity but at the same time respecting their privacy. Masking all the *position* attribute would deliver similar information to a mechanism showing that a user is present in a document. If additional information is given, for instance the *level* or the *length*, the effect of user changes could be computed for the document level. Although such an operation is correctly handled by the proposed mechanism, we believe that the previously mentioned cases are far more interesting and challenging and we focus on them only. Examples of ghost operations of this privacy level are:

gOp4= $< edit, null, [2], null, null, null >$.

gOp5= $< edit, 2, null, null, null, null >$.

gOp6= $< edit, null, null, null, null, user1 >$.

Visualising ghost operations through edit profiles

By means of an example we first show how various types of ghosts operations are visualised. Next, we provide details about the computation of awareness information.

Consider the case of some researchers authoring a research paper. For simplicity, we consider that the document being edited contains 4 paragraphs. Awareness by means of edit profiles for committed operations was presented in [20]. In this section, we show how this could be extended to provide a visualisation mechanism for uncommitted changes in the presence of ghost operations. Consider that users Mary, Bob and Tom work on the same version of the document and they generate concurrently the changes illustrated in Figure 3.

Mary wants to let the others know about the changes she did in the second paragraph (the title of the section is considered as a paragraph itself), but does not want at this point to reveal her identity for those changes. She therefore generates 8 ghost operations $gOp_{1Mary}, \ldots, gOp_{8Mary}$ corresponding to the insertion of words 'of', 'individuals', 'of', 'interest', '(Computer', 'Supported', 'Cooperative', 'Work)'. For instance, the ghost operation for the insertion of the first word 'of' is $gOp_{1Mary} =< insert, 2, [2, 1, 7], 'of', [0, 0, 1, 2], null >$. Concerning the changes Mary did on paragraph 4, she wants to let the other users know that she is editing that paragraph but still needs some time to review what she has edited. She chooses to send her changes with a masked content, masked type and partially masked position (only the paragraph level node is given). However, she does not filter the length of her changes, providing in this way a measure of the changes that she made. She generates 17 ghost operations $gOp_{9Mary}, \ldots, gOp_{25Mary}$ corresponding to the insertion of character ',', insertion of word 'SubEthaEdit', deletion of words 'or', 'software' and 'development', insertion of words 'synchronous', 'pair-programming', deletion of word 'However' and insertion of words 'Although', 'real-time', 'applications', 'appear', 'to', 'be', 'rather', 'promising'. For instance, $gOp_{9Mary} =< edit, 3, [4], null, [0, 0, 0, 1], Mary >$.

Bob decides to send unmasked to the other users the changes that he made in paragraph 2 of the document. He therefore generates 6 ghost operations $gOp_{1Bob}, \ldots, gOp_{6Bob}$ corresponding to the insertion of words 'business', 'and', 'Not', 'surprisingly,', deletion of character 'T' and insertion of character 't'. For instance, $gOp_{1Bob} =< insert, 2, [2, 1, 4], 'business', [0, 0, 1, 8], Bob >$. As he wants to review the changes he did in paragraph 3, he will mask their content, but send all the other information required. He generates two ghost operations gOp_{7Bob} and gOp_{8Bob} corresponding to the insertion of sentences 'Synchronous authoring tools, also referred to as real-time collaborative authoring systems, imply that changes made by one user are im-

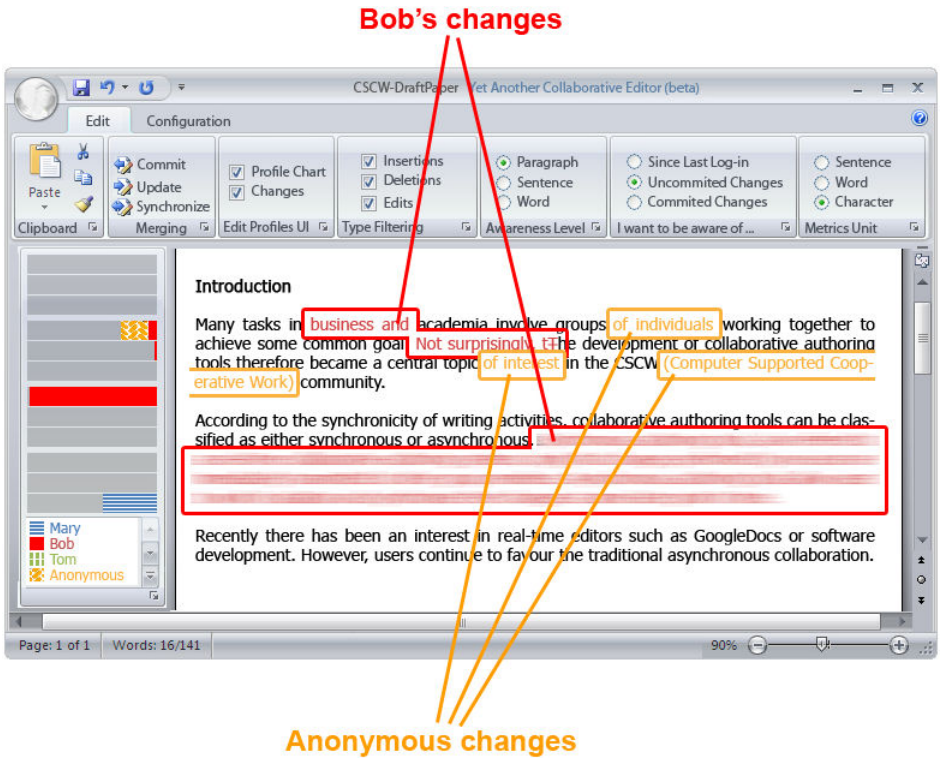

Figure 4. Edit profiles in the presence of ghost operations.

mediately transmitted to other group members.' and 'Asynchronous authoring tools allow users to work in isolation and synchronise their changes at some later point in time'. For instance, $gOp_{7Bob} =< insert, 1, [3, 2], null, \{0, 1, 25, 184\}, Bob >$.

Tom just started working on the document, he did just a small change. He just wants to let the others know that he has started working on the document and therefore he masks all information about his changes except his identity. He therefore generates the ghost operation $gOp_{1Tom} =< edit, null, null, null, null, Tom >$.

Figure 4 represents the edit profiles in the presence of the above described masked concurrent changes as seen by a fourth researcher who just opened the document containing the paper. The researcher wants to be informed about the uncommitted changes of his colleagues presented by means of a profile chart (left side of the figure) that gives him a quick overview of the changes along with the changes themselves in the document. He therefore selected the *ProfileChart* and *Changes* options. He has selected to see all types of changes, i.e. insertions, deletions and edits presented at the level of paragraph, changes being measured in characters. The profile chart includes three bars for each paragraph, showing in a top-to-down order the number of insertions, deletions and edits made on that paragraph. Changes of one user are identified by a unique colour, in the lower left side a legend of users with their associated colours is presented. Due to black and white printing, patterns are used in addition to colours. From the profile chart, we can deduce that no changes were done in the first paragraph, that the changes in the second paragraph were made by Bob (insertions and deletions) and an anonymous user (insertions), that Bob contributed also by inserting content in the third paragraph and that Mary edited the last paragraph. We can also see that another user Tom is active in the document, but no information about the changes he made is provided. Inside the document, changes made by users are included if their content and position were not masked. If the content of changes was masked but their positions and lengths were specified, the changes are blurred.

Visualising the changes superimposed on the document is an optional feature and can be deactivated by the users. Moreover, if users want to work without being disturbed about uncommitted changes, they can choose to visualise only the committed changes, or only their local changes. We expect that the most used work mode settings of our proposed awareness mechanism are the ones where edit profiles are chosen to be displayed on real-time and merged content only presented on request. However, the user can work in a complete isolation by deselecting the *ProfileChart* and *Changes* options.

As we can see, our system offers users the possibility of working in complete isolation as in the case of traditional multi-synchronous communication or to view in real-time the changes made in the system. Moreover, our system offers users the flexibility of filtering, according to their privacy, the information about their changes that are transmitted to their collaborators. If none of the users filters the transmitted changes and all users select the *Changes* option to integrate

user changes as soon as they occur, our system simulates the functionality of a real-time system enhanced with edit profiles.

Awareness computation using ghost operations

In this subsection, we consider the above examples of ghost operations and explain how our mechanism computes awareness information. We assume that the operations arrive at a collaborator's site and are about to be evaluated.

Operations like $gOp_{1Bob} =< insert, 2, [2, 1, 4], 'business', [0, 0, 1, 8], Bob >$, belong to the first level we introduced, where no privacy issues are taken into consideration. For the computation of awareness, the document node where the operation will be applied is found first. An *opValue* is then computed for each of the defined metrics and its value, together with information about the type of the operation and the user who created it, is attached to the node by being added to the corresponding *nodeValue*. The resulting *nodeValues* are visualised through the edit profile. Their values are shown in the chart at the bar referring to the node and the operation's type. The changes are also superimposed on the text. The pattern and colour used in the chart and in the text respectively show the user who made the change.

A procedure similar to the above one is followed when the ghost operation has the user attribute masked as in operation $gOp_{1Mary} =< insert, 2, [2, 1, 7], 'of', [0, 0, 1, 2], null >$. It only differs at the step where the computed values are attached to the node. Since the user is not known, the values are attached to the node and marked as being made by user "Anonymous". This "new" user is introduced to hold all the changes made from users who want to keep their anonymity.

If the ghost operation arriving at a site provides no content, then it is impossible to annotate the document. If, however, adequate information is given about the effect of the operation and the position where it is applied, then our mechanism informs the user about the details of the change. For instance, the effect of operation $gOp_{7Bob} =< insert, 1, [3, 2], null, [0, 1, 25, 184], Bob >$ can easily be evaluated since the length is provided. Additionally, the level can be used to check whether the position is partially masked or not. In this operation, we can easily see that the position is not masked, since the operation informs us about a new sentence that will be inserted at the third paragraph in position 2. This means that the exact node where the operation will be applied is given. All of the above information will be provided through the edit profile, but the document will not be annotated with the correct text, since the content is not given. To illustrate, however, that the position and the length of the change is known, the document will be annotated with blurred text.

If the content is masked and the combination of level and position show that the position is partially masked, then the exact node where the operation is applied cannot be found. Using the given part of the position, an approximation of the exact position can be found, i.e. the closest ancestor of the changed node is found. For instance, the operation $gOp_{9Mary} =< edit, 3, [4], null, [0, 0, 0, 1], Mary >$ is of level character, but only the paragraph is given in the position element. Paragraph 4 is the deepest we can reach in the hierarchical document and hence we assign to this node all the computed information. Since we do not know exactly where the changes are made in the paragraph, the document is not annotated at all. This helps the user to distinguish this case from the previous one.

CONCLUSIONS AND FUTURE WORK

We presented an awareness mechanism which, in the light of privacy issues, computes real–time awareness in multi-synchronous collaboration. Users can choose through a set of privacy levels the one that best fits their role, collaborative task, current needs and situation. The detail of information sent to other users about their uncommitted changes is defined by these levels. Based on the privacy level, various filters are applied to the original operations to mask some of the operation attributes and create ghost operations sent to other users. Upon receipt at a remote site, ghost operations are specially handled by our awareness mechanism to extract the maximum available information. The computed awareness information is finally visualised by means of edit profiles which enable users to have a quick overview of the hot areas that contain concurrent changes done by various users.

We believe that the examples used in this paper constitute a representative, but by no means exhaustive, list of the most common activities and privacy concerns that users have when involved in collaborative tasks. In the future, we intend to conduct user studies to analyse whether our awareness approach achieves better improvements over traditional multi-synchronous collaboration and check whether there are any other privacy levels that we need to take into consideration to extend our mechanism towards satisfying more users in various collaborative situations.

Another direction we want to follow is to automatically generate the filtering mechanism according to a trust metric showing a measure of how group members are trusted by other group members. Operations generated by a user will be therefore filtered and sent to other users according to the level of trust the user has in other users.

REFERENCES

1. Google Docs & Spreadsheets. *Create and share your work online*, 2008. http://docs.google.com.
2. Subethaedit. *Collaborative text editing. Share and Enjoy*, 2008. http://www.codingmonkeys.de/subethaedit/.
3. B. Berliner. CVS II: Parallelizing Software Development. In *Proceedings of the USENIX Winter Technical Conference*, pages 341–352, Washington, District of Columbia, USA, January 1990.
4. M. Boyle and S. Greenberg. The Language of Privacy: Learning from Video Media Space Analysis and Design. *ACM Transactions on Computer-Human Interaction*, 12(2):328–370, 2005.

5. B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato. *Version control with Subversion*. O'Reilly & Associates, Inc., 2004.

6. P. Dewan and R. Hegde. Semi-synchronous conflict detection and resolution in asynchronous software development. In *Proceedings of the European Conference on Computer-Supported Cooperative Work (ECSCW'07)*, pages 159–178, Limerick, Ireland, September 2007. Springer London.

7. P. Dourish. The parting of the ways: Divergence, data management and collaborative work. In *Proceedings of the European Conference on Computer-supported Cooperative Work (ECSCW'95)*, pages 215–230, Stockholm, Sweden, 1995. Kluwer Academic Publishers.

8. P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW'92)*, pages 107–114, Toronto, Ontario, Canada, 1992. ACM Press.

9. C. A. Ellis, S. J. Gibbs, and G. Rein. Groupware: some issues and experiences. *Communications of the ACM*, 34(1):39–58, 1991.

10. G. Fitzpatrick, P. Marshall, and A. Phillips. Cvs integration with notification and chat: lightweight software team collaboration. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW'06)*, pages 49–58, Banff, Alberta, Canada, 2006. ACM.

11. C. Gutwin. *Workspace Awareness in Real-time Groupware Environments*. Ph.D. Thesis, Department of Computer Science, University of Calgary, Calgary, Canada, 1997.

12. C.-L. Ignat and M. C. Norrie. Customizable Collaborative Editor Relying on treeOPT Algorithm. In *Proceedings of the European Conference on Computer-supported Cooperative Work (ECSCW'03)*, pages 315–334, Helsinki, Finland, September 2003. Kluwer Academic Publishers.

13. C.-L. Ignat, G. Oster, P. Molli, and H. Skaf-Molli. A Collaborative Writing Mode for Avoiding Blind Modifications. In *Ninth International Workshop on Collaborative Editing Systems, GROUP'07*, Sanibel Island, Florida, USA, November 2007.

14. C. McDowell, L. Werner, H. E. Bullock, and J. Fernald. The impact of pair programming on student performance, perception and persistence. In *Proceedings of the International Conference on Software Engineering (ICSE'03)*, pages 602–607, Portland, Oregon, USA, 2003. IEEE Computer Society.

15. P. Molli, H. Skaf-Molli, and C. Bouthier. State Treemap: an Awareness Widget for Multi-Synchronous Groupware. In *Proceedings of the Seventh International Workshop on Groupware (CRIWG'01)*, pages 106–114, Darmstadt, Germany, September 2001. IEEE Computer Society.

16. P. Molli, H. Skaf-Molli, and G. Oster. Divergence awareness for virtual team through the web. In *Integrated Design and Process Technology (IDPT'02)*, Pasadena, California, USA, June 2002.

17. S. Noël and J.-M. Robert. Empirical study on collaborative writing: What do co-authors do, use, and like? *Journal of Computer Supported Cooperative Work*, 13(1):63–89, 2004.

18. J. S. Olson, G. M. Olson, L. A. Mack, and P. Wellner. Concurrent Editing: The Group's Interface. In *Proceedings of the International Conference on Human-Computer Interaction (INTERACT'90)*, pages 835–840, Amsterdam, Netherlands, 1990. Elsevier Science Publishers, North-Holland.

19. S. Papadopoulou, C.-L. Ignat, G. Oster, and M. Norrie. Increasing Awareness in Collaborative Authoring through Edit Profiling. In *Proceedings of the IEEE Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'06)*, pages 1–9, Atlanta, Georgia, USA, November 2006. IEEE Computer Society.

20. S. Papadopoulou and M. Norrie. Shadow Document Sets for Synchronously-Aware Asynchronous Collaboration. In *Proceedings of the IEEE Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'07)*, New York, New York, USA, November 2007. IEEE Computer Society.

21. S. Papadopoulou, E. Reuss, and M. C. Norrie. A user study of edit profiles in collaborative authoring systems. In *Proceedings of the International Symposium on Collaborative Technologies and Systems (CTS'08)*, Irvine, California, USA, May 2008.

22. R. Robbes and M. Lanza. A change-based approach to software evolution. *Electronic Notes in Theoretical Computer Science*, 166:93–109, 2007.

23. A. Sarma, Z. Noroozi, and A. van der Hoek. Palantìr: Raising Awareness among Configuration Management Worspaces. In *Proceedings of the International Conference on Software Engineering (ICSE'03)*, pages 444–454, Portland, Oregon, USA, May 2003. IEEE Computer Society.

24. J. Tam and S. Greenberg. A Framework for Asynchronous Change Awareness in Collaborative Documents and Workspaces. *International Journal of Human-Computer Studies*, 64(7):583–598, July 2006.

25. J. R. Tam. Supporting change awareness in visual workspaces. Master's thesis, Department of Computer Science, University of Calgary, Alberta, Canada, 2002.