A Flexible Distribution Service for a Co-authoring Environment on the Web

Sonia Mendoza  
Laboratoire LSR-IMAG  
Grenoble, France  
Sonia.Mendoza@imag.fr

Dominique Decouchant  
Laboratoire LSR-IMAG  
Grenoble, France  
Dominique.Decouchant@imag.fr

Alberto L. Morán  
Facultad de Ciencias, UABC  
Ensenada, B.C., Mexico  
almys@uabc.mx

Ana María Martínez Enríquez  
Departamento de Ingeniería Eléctrica,  
CINVESTAV-IPN, D.F., Mexico  
ammartin@cinvestav.mx

Jesus Favela  
Departamento de Ciencias de la Computación,  
CICESE, Ensenada, B.C., Mexico  
favela@cicese.mx

Abstract

The PIÑAS platform provides support to collaboratively and consistently produce shared documents in the Web environment. Such documents may include possible costly multimedia resources, whose management raises important issues due to the constraints imposed by Web technology. In this paper, we present a flexible service for distributing shared Web documents across authoring group sites. To carry out distribution, our approach takes into account the current organization of the involved sites, the access rights granted to the co-authors and the site storage capabilities. Scenarios are used to motivate the need for robust mechanisms for the management of shared Web documents and their resources, and to illustrate how our approach addresses these issues.

1. Introduction

In the CSCW field, distribution concerns the way in which the shared objects of a groupware application are distributed across the involved sites. Distribution strategies mainly differ in three aspects: 1) the number of object copies available in an application at a given moment; 2) the object representation at the local site; and 3) the mobility of objects around the network. These aspects can influence the work style of an authoring group. As a collaborative task evolves, the group work style could change in order to satisfy both individual and collective needs.

Flexible distribution support was identified (e.g. [4]) as an important issue in groupware design and development. This support refers to a framework capability for allowing the application developers to implement different distribution strategies, according to their own requirements. Most frameworks only support one distribution strategy, which applies to all shared objects of a single application. A few frameworks (e.g. [4] [7] [9]) take flexibility into account for their distribution support. However, they mainly support synchronous work. Current works on the Web (e.g. [13]) focus on providing asynchronous support for document co-authoring. Nevertheless, these solutions just offer one distribution strategy.

Also, the Web technology imposes constraints (e.g. client site and user identification is not yet integrated into the access mechanism) raising important issues of distributed object management.

We propose a flexible service for distributing shared Web documents across authoring group sites. Distribution strategies apply not only to different applications, but also to individual objects of a single application. To distribute documents, we take into account the current organization of the involved sites, the site storage capabilities and the access rights granted to the co-authors. These aspects have not been previously addressed by other frameworks in a comprehensive way. Concurrency control issues are not explicitly addressed here [8].

The remainder of this paper is organized as follows. After presenting the architecture of the PIÑAS platform, Section 3 introduces the data model of the Web document entity. In Section 4, we describe the principles to organize authoring group sites. Section 5 describes our approach for sharing and distributing Web documents across co-author sites. Section 6 discusses related work and, finally, Section 7 presents conclusion and future work.
2. The PIÑAS platform

The PIÑAS platform [2] provides support to collaboratively and consistently produce shared documents in the Web environment. It is based on a client/server architecture, whose server side consists of three main layers (see bottom block of Fig. 1). The bottom layer offers a set of HTTP-based proprietary communication functions. The middle layer contains the collaborative entities offered by the platform, as well as their management services. And finally, the top layer provides a synchronous API (Application Program Interface) to access these services.

![Figure 1. Architecture of the PIÑAS platform](image)

The PIÑAS platform manages different types of entities, such as Author, Document, Project, Application and Session, which are essential to the collaboration support. Author entity manages information allowing to identify a collaborator at the system and social levels. Author identification allows to grant access rights to a collaborator, to determine his/her contributions and to coordinate his/her actions. Document entity represents a Web document shared by an authoring group. It can contain other entities such as fragments and multimedia resources. Project entity maintains the Author and Document entities involved in a collaborative endeavor. Application entity allows authors to know the status (e.g., presence, availability, etc.) of other authors/documents and to collaboratively work on a document. Finally, Session entity handles information about the Author and Document entities (i.e., Project entity) associated to an Application entity.

The top block of Figure 1 separates the applications interacting with the platform through the synchronous API (server side) from those interacting by means of the asynchronous one (client side). Applications use the asynchronous API when they lack of communication support to interact with the platform services. The assistant module carries out all communications with the synchronous API, while the entity modules at the client side accomplish function calls of the corresponding modules at the server side.

The PIÑAS platform supports three types of applications: PIÑAS-based groupware applications (e.g., AllianceWeb), standard Web browsers (e.g., Netscape) and dedicated system or environment applications (e.g., an inference engine). Particularly, standard Web browsers have limited access to the platform entities and services.

3. Data model of the Web document entity

A key component of our platform is the Web document entity, which consists of one or more fragment entities (see Fig. 2). In turn, a fragment entity can refer to multimedia resource entities. Two different documents can not maintain the same total sequence of fragments, but they can share subsequences. Conversely, two different fragments of the same document can contain common resources. A document has to hold at least one fragment, while a fragment may not include any resources. A fragment can belong to a document only once. Similarly, a resource can be member of a fragment only one time.

![Figure 2. Data model of the Web document, fragment and resource entities](image)
functional aspects such as distribution, consistency, access control and persistency. This conceptual separation allows to manage non-functional aspects of a Web document independently of those of its fragments and resources. The distribution component maintains location information about other replicas and proxies in the platform. The consistency component contains the nature of the replicas for controlling concurrency (i.e., master or slave replicas) and references the other master and slave ones. The access control component keeps the definition about the access rights granted to co-authors on the entity. Finally, the persistency component holds information about its physical name and location in the underlying storage system.

An entity consists of a set of properties and a body. In turn, a body contains a sequence of fragment references. An entity comprises a HTML, XHTML or XML/DTD description about a sequence of fragment references. Similarly, a fragment body holds a description about a sequence of resource references. At creation time, a document just maintains one fragment, whose body can be initialized with a possible empty description. A resource body can contain a GIF/JPEG image, a basic/tone audio or a MPEG/pointer video, etc.

Figure 3. Representation of a shared entity

Data representation at the local site is an important issue to take into account when implementing distribution strategies. The data model supports two types of representations: proxy and real entity. These representations permit the application developers to respectively implement centralized or replicated objects. Representing an entity in proxy form allows to remotely referring to it. Conversely, creating a real entity allows it to be locally referenced. The main difference between these representations resides on the entity body: a real entity holds a body, while a proxy is a descriptor that establishes a reference to the body.

An author can own several Web documents, which are organized into a documentation base. In fact, a documentation base may be empty at a given moment (e.g., at author registration time). Different author's bases can contain common documents in order to allow document sharing. Private documents are stored in the owner author's sites, while shared documents also are distributed across those of his colleagues. Besides, in order to provide co-authors with high availability of information, a documentation base can be replicated in different sites as explained below.

4. Principles to organize co-author sites

To carry out distribution of co-authored documents, we distinguish a working site from a storage site [8]. A working site is an end-user computer, running PIÑAS-based applications and possibly standard Web applications, where an author consults and modifies documents. A storage site is a Web server host, running PIÑAS-based services, where the author's documents are stored. Every author has a set of working sites and a set of storage sites at his disposal. Several authors can share a working and/or storage site. Moreover, a single site can act both as a storage or as a working site to allow autonomous work.

We use the concept of nomadic mode work based on the multi-site author principle stated above. We define it as the capability for an author to establish alternative connections to/from different pre-specified sites as he/she moves. Thus, an author is able to transparently work on shared documents from different sites, and dynamically retrieve his/her working environment at any moment.

Based on the previous concept, we specify a first level of nomadic work features by relating a working site to a set of storage sites that can handle author's requests. In addition, author/site identification is seamlessly integrated into the access mechanism. In this way, when an author sends a request to a storage site, the corresponding service first identifies him to verify whether he is a valid author in the platform. Author identification is carried out by testing three parameter groups, which are listed in decreasing priority: 1) the author identifier, 2) the working site name and the author login, and 3) the author social name. An author is validated if any of the three parameter groups tests true.
The following scenario illustrates the previous concepts (see Fig. 4). Authors, Ray, Beth, Kurt and Saul, dispersed around the world, have several working and storage sites at their disposal. Ray can produce and consult documents on the progreso.mx, koln.de and versailles.fr working sites, and download and upload them from the cancun.mx, bavaria.de and louvre.fr storage sites. Ray’s sites are located in different countries, Mexico, Germany and France to allow him to work in nomadic mode. Based on his current physical location, e.g. progreso.mx, he can access documents from a close site, e.g. cancun.mx, using a more reliable environment (e.g. LAN).

Such an organization provides Ray with high availability of his documents by distributing them across several sites. As depicted by the gray shadow, he can rely on a platform that automatically distributes his documents to make them available during a working session and to persistently store them. In addition to these features, the underlying platform ensures that Ray’s documents will be updated and kept consistent, even in an unreliable environment (e.g. internet). Also, different authors’ working sites, e.g. Ray’s site, versailles.fr, and Beth’s site, orsay.fr, can download and upload documents from a common storage site, e.g. louvre.fr. In this way, several authors located in relatively close places (e.g. in the same LAN) can share storage capabilities (e.g. storage). Finally, the tulum.mx site can simultaneously act as working and storage site to allow Saul to work in an autonomous way.

5. Flexible distribution service

In this section, we first characterize the distribution of documents across co-author sites. Then, we describe the phases of our distribution protocol. Finally, we explain how a document can be shared among several authors, and how its multimedia resources are distributed across the involved sites.

5.1. Distribution of Web document entities

At creation time, a document is private and belongs to its creator. In order to make it available during a working session and to persistently store it, such a document would be distributed across several sites. For example, allow us to consider a scenario in which Ray produces the report document on the progreso.mx working site (see Fig. 5). As mentioned earlier, Ray has the cancun.mx, bavaria.de and louvre.fr storage sites and the progreso.mx, koln.de and versailles.fr working sites at his disposal. Thus, a replica of the report document is permanently created at each Ray’s storage site. Such an organization provides Ray with high availability of his documents. Additionally, another replica is temporarily created at his current working site, e.g. progreso.mx, whenever Ray aims at consulting or modifying the document contents. In this way, rapid feedback at the user interface level is achieved. The report document can be also distributed across the koln.de and versailles.fr working sites.

Figure 4. A typical organization of co-author sites

Figure 5. Distribution of the report document
Identification and location of co-author sites are important issues when designing a distribution service. To overcome them, we define the author definition entity [2]. For every author, it holds the sets of storage and working sites available to him, the set of relations among these sites and some configuration data. An author definition is replicated in all storage and working sites at the corresponding author’s disposal. Following the evolution of such an entity (i.e. site creation, modification or destruction), the replicas are regularly updated. Each time an author starts a session at one of his working sites, the platform determines, from his author definition, the storage sites from where documents can be downloaded. During this connection phase, the author definition at the current working site can be updated.

5.2. Distribution protocol

The distribution service relies on a two phase protocol to carry out document distribution across the co-author sites. The first phase concerns communication between a working and a storage site, while the second phase involves communication among storage sites. To describe these phases, we present a scenario for treating the request about the creation of the report document, as mentioned above.

5.2.1. Phase 1: request for creating a document. In this phase, author Ray working at the progreso.mx site asks a groupware application (e.g. AllianceWeb [2]) to create the report document (see Fig. 6). Subsequently, the application inspects the local author definition to identify the storage site(s) that can handle its requests (step 1). According to the first scenario (c.f. Section 4), the progreso.mx working site is related to the cancun.mx storage site. In this way, the application forwards the request to the distribution service at the cancun.mx site (step 2). In turn, such a service creates a local replica of the report document and sends an acknowledgment to the application (step 3). Then, a local copy is temporarily created in the user’s cache at the progreso.mx site to allow Ray to immediately manage and/or work on the report document (step 4).

5.2.2. Phase 2: distribution of a document entity. In this phase, the service at the cancun.mx site is responsible for distributing the report document. To carry out this task, it first examines the local author definition, belonging to Ray, in order to determine the storage sites the document is to be distributed to (step 5). According to the first scenario (c.f. Section 4), the storage sites bavaria.de and louvre.fr are also available to Ray. The service at the cancun.mx site determines the type of representation (e.g. proxy or replica) to be created in those sites. By default, a temporal replica (i.e. only valid for a session) is created at a working site, while either a persistent replica or proxy can be created at a storage site. In this way, the service at the cancun.mx site asks its peers at the bavaria.de and louvre.fr sites to create a replica of the report document (step 6). Finally, they create a replica at their sites and then send an acknowledgment to their peer at the cancun.mx site (step 7).

Figure 6. Phases of the distribution protocol

Notice that the different replicas do not have to keep explicit references among each other, as the author definition holds the needed information. In other words, the distribution component of the report replica at the louvre.fr site does not have to hold an explicit reference to the replica at the bavaria.de site and vice-versa. Similarly, they do not need to know about the replica at the cancun.mx site and vice-versa, as the local author definition maintains the set of storage sites where persistent replicas can be saved. Like persistent replicas, the distribution component of the temporal replica at the progreso.mx refers to the local author definition, instead of holding a reference to the persistent replica at the cancun.mx site. In this way, if the network connection between the progreso.mx and cancun.mx sites fails, or the cancun.mx site fails, then the louvre.fr site can handle author requests concerning the report document.
5.3. Sharing Web document entities

To allow several authors to concurrently work on a shared document, a document owner has to specify the corresponding authoring group and subgroups, break up a document into several fragments, and grant access rights to individual co-authors and subgroups on the fragments and resources. AllianceWeb [2] is the PIÑAS-based application that supports both document fragmentation and access right granting. A fragment can be recursively divided to form new ones, according to the document logical structure. The number of fragments can vary from one document to another, depending on the requirements of the authoring group. Thus, authors can access document fragments in accordance with their access rights.

To exemplify these concepts, let us suppose that Beth, Kurt and Saul are Ray's colleagues. As Ray decides to share the report document with Beth and Saul, he creates the corresponding authoring group. Then, he breaks up the report document in three fragments, f1, f2 and f3, and grants access rights to the fragment members. Let us consider that Beth and Saul are allowed to read fragment f1, while Ray may write it. Concerning fragment f2, Saul is permitted to write it, while Beth and Ray may read it. Finally, Saul does not have any access rights on fragment f3, while Beth and Ray may review and write it, respectively.

As mentioned in Section 3, a fragment can refer to multimedia resources by means of hyperlinks. Thus, the referenced resources can inherit the access control definition of the referencing fragment. However, the access rights on its resources can be redefined in order to increase the co-authoring concurrency on a fragment. Let us suppose that some fragments of the report document refer to multimedia resources. Fragment f1 references to the image and audio resources, while fragment f3 includes a video resource. As for fragment f2, it does not refer to any. In this way, the image and audio resources inherit the access control definition of f1, while the video inherits that of f3. Therefore, only Ray is authorized to modify the image and audio resources, while Beth and Saul may visualize them. As for the video resource, Beth and Ray can respectively review and modify them, while Saul does not have any access rights. To increase authoring concurrency on fragment f1, Ray redefines the access rights on the image and audio resources. Thus, for instance, Ray still might modify fragment f1, while Beth would be allowed to change the image and Saul might modify the audio. Even more, Ray might not have any access rights on the audio.

5.4. Distribution of resource entities

Before starting redistribution of the report document, an initialization phase has to be performed across group sites. In this phase, the author definition of each member is exchanged and persistently stored in the working and storage sites of the group members. Replicas of Ray's author definition are distributed across the sites of Beth and Saul and vice-versa. Thus, each member can have an alternative connection to another storage site and work with shared and distributed documents in a dynamic and transparent way.

As soon as redistribution took place, the report document is available at Ray's storage sites (i.e. cancun.mx, bavaria.de and louvre.fr) as well as at those of Beth (i.e. cancun.mx and louvre.fr) and Saul (i.e. cancun.mx and tulum.mx). Although Ray, Beth and Saul share common storage sites (i.e. cancun.mx) just one replica of the report document is held in those sites. To collaboratively co-author the report document, a replica can be temporarily created at Ray's working sites (i.e. progreso.mx, koln.de and versailles.fr) and at those sites available to Beth (i.e. merida.mx and orsay.fr) and Saul (i.e. tulum.mx).

![Figure 7. Distribution of shared resources](image)

Like documents, replicas of multimedia resources are distributed across the group storage sites. However, proxies can be created at some sites in order...
to save storage space. Decisions about creating either a replica or a proxy at a storage site also depend on the access rights granted to the corresponding author on the resource (see Fig. 7). For example, concerning the image, replicas are distributed across the group storage sites (i.e., cancun.mx, tulum.mx, louvre.fr, and bavaria.de), as the group members have access rights on them. Conversely, proxies for the audio would be distributed across Ray’s sites (i.e., cancun.mx, bavaria.de, and louvre.fr) because he does not have any access rights. However, as Berh shares the cancun.mx and louvre.fr storage sites with Ray and she does have access rights on them, replicas rather than proxies are created at those sites. Similarly, proxies for the video would be distributed across Saul’s storage sites (i.e., cancun.mx and tulum.mx) as he does not have any access rights. Nevertheless, as Berh and Ray share the cancun.mx storage site with Saul and they do have access rights on the video resource, a replica rather than a proxy is created at this site.

By distributing resources according to the author access rights, replicas of possibly costly multimedia resources are created at the storage sites, whose users have access rights on them. Otherwise, a proxy is persistently kept in order to save storage space. Additionally, replicas or proxies for resources can be distributed across working sites. A replica can be temporarily created at his/her current working site, if the author authorizes him to access the resource. In this way, the author can be provided with rapid feedback at the user interface. Instead, a proxy is created. Moreover, when an author (e.g., Ray) has access rights on costly resources (e.g., video) but he can not actually access them due to device constraints (e.g., a PDA has a limited storage space), temporal proxies rather than replicas are created at his current working site (e.g., progreso.mx), even if replicas are held in his storage sites (e.g., cancun.mx, bavaria.de, and louvre.fr).

6. Related work

Most groupware frameworks and toolkits support one distribution strategy, which applies to the whole application. For instance, Rendezvous [6] provides the centralized strategy, while GroupKit [11] and COAST [12] are based on the replicated one. Suite [3] offers a hybrid strategy, which centralizes the shared data and replicates the application programs. Only few frameworks, such as GEN [9] and DreamObjects [7], provide flexible support for distributing shared objects, which allows the application developer to implement different distribution strategies. By default, GEN offers implementations for centralized and replicated strategies. A shared object can be represented as proxy or replica in order to respectively implement a centralized or replicated object. By means of the open implementation technique, this platform also allows the developer to specify how a shared object is distributed and maintained consistent. As the developer can modify the implementation, he/she has to be aware of the underlying concepts and protocols.

DreamObjects provides replicated, asymmetric and adaptive strategies. In contrast to the centralized strategy, the asymmetric one does not require a well-known server. The adaptive strategy dynamically changes the distribution characteristics of a shared object according to the user working style. The developer can specify the distribution and consistency strategies of a shared object at runtime. By applying the substitution principle, this platform achieves transparency for the developer.

Although GEN and DreamObjects focus on providing several distribution strategies, they are mainly concerned with synchronous work support. Consequently, they do not address other important issues (e.g., persistency) that arise from the need to support disconnected work. Additionally, the document management field is also interested in designing and developing systems that support asynchronous work. Examples of document managers are Lotus Notes [13], Bayou [5] and DOORS [10]. Even though these systems take persistency into account, they are constrained to a replicated strategy.

Except for Lotus Notes, none of the previous systems is adapted for the Web. Current research efforts, such as WebDAV [13] and BSCW [1], focus on providing functions for accessing and publishing Web documents on remote servers. Users are able to lock documents, to get and modify them, and finally to send them back to the server. However, these functions are only designed for a hybrid strategy, which makes them difficult to maintain document consistency within an unreliable environment (e.g., Internet).

Our platform proposes to go a step further by proving a flexible service for distributing shared Web documents. Our approach takes into account important issues, such as storage capabilities, access control and current organization of the co-authors sites, arising from the need to support collaborative work in disconnected mode on the Web. These issues have not been previously addressed by other groupware frameworks and toolkits in a comprehensive way.
7. Conclusion and future work

In order to support collaborative work in disconnected and degraded mode on the Web, we focused on the multi-site author principle, which relates an author with several working and storage sites. Additionally, associations among working and storage sites can be established to provide a first level of nomadic work. In this way, authors can dynamically and transparently transfer their working environment from one place to another. Based on this principle, we proposed a flexible service for distributing shared Web documents across an authoring group sites. Distribution is carried out according to the current organization in terms of working and storage sites, the site storage capabilities and the author access rights. Documents and resources can be represented both in replica and proxy form.

There are still many open issues. We intend to provide flexible support for managing consistency of shared documentation bases. Our current approach allows two or more authors to potentially play roles that authorize them to concurrently modify a single resource contents. For controlling concurrency, a pessimistic strategy based on one master and many slave copies is used. However, when resources are shared among several documents produced by different authoring groups, this strategy seems restrictive. We also aim to support automatic detection of neighboring sites. Thus, a working site can get documents from the closest storage site of those sites at its disposal. Another important issue concerns the adaptive support for propagating state changes.

8. Acknowledgments

This work is supported by ECOS/ANUIES project M03M01.

9. References


