Raphael de Oliveira Santos

A Comprehensive Environment for Collaborative Web Browsing – Pragmatic Specification and Development Approach

Vitória - ES, Brasil

22 de Maio 2009

Raphael de Oliveira Santos

A Comprehensive Environment for Collaborative Web Browsing – Pragmatic Specification and Development Approach

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo para obtenção do título de Mestre em Informática.

Orientador:
Prof. Dr. Magnos Martinello

Orientadora:
Prof^a. Dr^a. Roberta Lima Gomes

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA
CENTRO TECNOLÓGICO
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória - ES, Brasil

22 de Maio 2009

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Santos, Raphael de Oliveira, 1984-

S237c A comprehensive environment for collaborative web browsing: pragmatic specification and development approach / Raphael de Oliveira Santos. – 2009.

127 f. : il.

Orientador: Magnos Martinello.

Co-Orientadora: Roberta Lima Gomes.

Dissertação (mestrado) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Grupos de trabalho. 2. Browsers (Programa de computador). 3. Redes e Sistemas Distribuídos. 4. Modelagem de dados. I. Martinello, Magnos. II. Gomes, Roberta Lima. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004

Dissertação de Mestrado sob o título *"A Comprehensive Environment for Collaborative Web Browsing – Pragmatic Specification and Development Approach"*, defendida por Raphael de Oliveira Santos e aprovada em 22 de Maio 2009, em Vitória, Estado do Espírito Santo, pela banca examinadora constituída pelos doutores:

Prof. Dr. Magnos Martinello
Orientador

Prof^a. Dr^a. Roberta Lima Gomes
Orientadora

Prof^a. Dr^a. Patrícia Dockhorn Costa
Examinador Interno

Prof. Dr. Daniel Ratton Figueiredo
Examinador Externo

Abstract

The variety of ways by which people interact have dramatically changed in the recent years. The conception of new teamwork paradigms has encouraged the creation of innovative collaboration technologies. This fact has led to the dynamic collaboration scenarios faced nowadays. Amongst these recent paradigms, there is one called *collaborative web browsing* (co-browsing) paradigm. Collaborative systems that implement such paradigm provide a useful way for virtual groups to share information through the web.

However, the common set of features in these tools is not enough to offer a more face-to-face-like browsing experience. To fill this gap, this work presents a novel collaborative web browsing proposal, which aims at integrating notably three important characteristics. Firstly a flexible management of sessions, which allows the involved participants (*i*) to contribute freely or hierarchically to the teamwork, but also (*ii*) to prevent confusion or waste of work effort. The second characteristic refers to the maintenance of a shared production spaces for co-browsing sessions. At last, providing efficient communication facilities. In this case, such efficient communication are accomplished by annotations (i.e.: draw and text note elements over co-browsed contents) and privileges negotiation facilities.

Additionally, the proposed environment analysis relies on a collaboration ontology as a reference model, which provides a well defined conceptualization and a common vocabulary about the collaboration domain. Another characteristic explored in this work involves an architectural solution for supporting suitable performance, regarding the system response time perceived by participants and how effective is the maintenance of awareness in the co-browsing sessions. Such challenges are met by the design of a lightweight distributed architecture and communication protocol. In order to demonstrate the feasibility of our approach, a prototype is developed and its performance issues evaluated.

Dedicatory

To Rosangela I.O. Santos and Olair J. Santos,
for being the best parents, ever.

Acknowledgements

During all this work long I have received such a valuable support from many partners, that I am absolutely certain that it would be impossible to obtain all these results without their help. Therefore this is a way the way I can say: "thank you so much, this work is also yours".

First of all, my advisors Magnos Martinello and Roberta Gomes. You have so much credit on all this, for guiding my steps and more important facing the problems that have appeared to me, without imposing any hierarchical condition or politics. This way, more than just coordinating this work and all the others we have conducted during my Master, we have became friends.

Great contributions have been provided by Felipe Oliveira and Julio Antunes to this work. These great friends have worked hard on LiCoB (the first version of OCEAN), since its conception. In this matter, Stanley Sperandio have helped a lot on producing the interface design for LiCoB and earlier for OCEAN too. In addition to that, special thanks to Felipe Oliveira and Bernardo Gonçalves, who also supported this work with several discussions and reviews about the collaboration ontology usage.

I'd like to thank professors Cesar Marcondes, Renata Guizzardi and Guillermo Hoyos-Rivera for the great help on discussing research questions and producing papers. In the same way, special thanks to Ramon Schwartz, who not only "pushed me"to the academic area but also worked a lot as a partner during the beginning of my Master, contributing a lot for my formation.

Another partners have contributed in punctual aspects of this work, in particular, Frederico Franzozi on managing reports, Aline Martins on providing references and Gondiberto Carvalho for English tips. Also, I would like to thank all volunteers that have participated on our experiments.

Last but not least, I have personal thanks to my family, my girlfriend Bruna Bertoldi, and my friends. These people have not participated actively in this work, however contributed a lot for my formation.

It's worth to mention that this work has been partially funded by FAPES/MCT/CNPq/CT-INFRA #36316008/2007, and Raphael O. Santos was supported by a master scholarship from CAPES/Brazil.

Contents

List of Figures

List of Tables

1 Introduction

The term "*collaboration*", according to Merriam-Webster[1] online dictionary is defined as the act of working jointly with others or together especially in an intellectual endeavor. This term has increasingly gained attention in computer systems, especially due to the growth of the Internet over the past years. In this scenario, new services denominated *collaborative* have been developed, becoming important tools for supporting groups and organizations to work together, in particular intermediated by the huge Internet infrastructure.

Considering this context, the focus of this work is oriented to collaborative web browsing paradigm, a subset of those collaborative services. Amongst the most relevant contributions of this work, we highlight a well-founded conceptualization of this collaboration paradigm, and a lightweight and flexible distributed system architecture. Moreover such proposed environment is implemented and evaluated by means of a proof-of-concept prototype.

This chapter introduces the motivation for the research in Section 1.1. Section 1.2 defines our specific objectives, situating the scope of this work. The proposed approach for dealing with collaborative web browsing is described in 1.3. Finally, the thesis structure is described in Section 1.4.

1.1 Motivation

A research field denominated Computer-Supported Cooperative Work (CSCW), has been recently created with the objective of gathering scientific and technological knowledge about computational solutions dedicated to foster collaboration among people. The evolution of web technologies and Internet infrastructure has been considered an important enabler for CSCW, favoring the emergence of new collaborative services. Such services have introduced innovative interaction ways among users, providing better results in team work, either for personal or professional purposes (1).

[1]Merriam-Webster Online Dictionary, available at: www.merriam-webster.com/dictionary.

Human beings have always worked and socialized in face-to-face groups. However, people no longer need to be in the same place to work together, especially in organizational context. Virtual groups can transcend distances, time zones, and organizational boundaries. As a result, the face-to-face nature of working relationships is changing dramatically, and a new business model gives rise to a novel organizational configuration, challenging the classical competitive market (2). Such emerging ubiquitous nature of business, organizations and even personal relations have attracted millions of users, demanding computational support for their newborn pervasive activities (3). In this matter, many applications aim to support such online collaborative work, for instance, email, conferencing (chat, audio or video), virtual workspaces and collaborative document writing. Among these collaborative solutions, we have the collaborative web browsing paradigm.

The collaborative web browsing paradigm consists of enabling web users to collaborate along their most common use of the web, that is, browsing web pages. The web (or www, from world wide web), is a publishing medium used to quickly disseminate information through the Internet (4), which basically represents a huge set of linked hypertext documents. In the traditional use of the web, users make solo tours through these documents, following the hyperlinks between them. The collaborative web browsing paradigm is, therefore, a proposal for embedding collaboration on this web tours, allowing users to jointly browse[2] through web documents.

Several application areas can take advantage of the collaborative web browsing paradigm. For example, web search, which is one of the most common online activities, is often undertaken in shared-computer context (5). Educators have also verified the added benefit of co-browsing for teaching as it fits nicely into the theory of constructivism, allowing students to learn by exploring and sharing their own ideas and knowledge (6). In fact, in different domains co-browsing systems have been commonly used as: (*i*) e-learning systems, to handle online lectures and presentations (1)(7)(8); (*ii*) helpdesk applications, to support users in guiding others through desired tasks (9); (*iii*) e-commerce environments, enabling users to recommend products or to negotiate purchases (10)(8); (*iv*) lightweight alternative for desktop sharing tools, to enable sharing of web-based content (11)(12); and recently (*v*) feeding social networks with browsing recommendations (13)(14).

[2]According to the Merriam-Webster online dictionary (www.merriam-webster.com/dictionary) the term "*browse*"refers to access a network by means of a browser. Considering the web context, "*web browsing*"is related to the act of accessing web published documents, and moreover, the terms "*web surfing*"and "*web navigating*"are commonly used synonyms for that.

1.2 Goals and Scope

Our main objective in this work can be summarized as follows: "*Propose an environment that fits users needs for collaboratively browsing the web with arbitrary purposes*".

This main goal led us to develop OCEAN, a co-browsing environment that was the result of our investigation in order to understand (i) the intrinsic characteristics of the collaborative web browsing paradigm, and (ii) how the current Internet infrastructure can support such pervasive collaborative activities.

1.2.1 Specific Objectives

This work addresses more than one research issue and in each phase of OCEAN development, leads to concentrate on specific problems. Therefore, our main goal can be refined into some specific goals, described as follows.

1. Provide *flexibility*, a property that we pursuit during all this work. It refers to design the collaboration environment adaptable to users' needs, as well as to the environment conditions.
2. Keep our work in accordance with well-founded collaboration theories, in an effort to produce broader and consistent results while avoiding mistakes during the whole development process.
3. Design an architecture capable of maintaining a high synchronization perception for users, while they collaboratively browse the web.
4. Adapt common features of traditional web browsing to this collaborative context. This way, users would feel more comfortable when using such environment.

1.2.2 Non-Objectives

Through these general objectives, we expect to contribute to the CSCW field. Even so, it is worth to clarify some issues that do not compose the objectives of this work. Such *non-goals* are described in the following:

- We do not intent to propose a general web collaboration environment, that would be used for any collaboration scenario. Environments like that usually integrate a broad number of collaborative functionalities. For instance, solutions like web-conference systems are in this category, providing instant messaging, audio/video calls, document editing, shared agenda, *etc*. Our scope in this work is strictly focused on the collaborative web browsing paradigm, since we have identified relevant issues that still need to be investigated in this field. Even though, our decision does not exclude the possibility of further integrating OCEAN in a broader collaboration environment.

- It is not our intention to develop a product or to offer a commercial solution. Our focus lies on performing a study identifying the major problems during OCEAN's development process, thus proposing and developing feasible solutions to overcome those problems.

- A common practice in CSCW is to evaluate developed solutions regarding some complex variables such as usability issues for individuals and groups (ease of use, effectiveness, efficiency, satisfaction) and the social and organizational impact of using OCEAN (15). Such qualitative evaluations could be obtained by formative studies(16), similar to the adopted on (5)(15)(11) and (6). However, at this stage we are just interested on more basic questions about our proposal, for instance, we worried like to evaluating OCEAN's specification and architectural design. Other evaluation aspects are also relevant and will be considered in the perspectives of future work.

- This work is not intended to be just an implementation guide of OCEAN, for instance, presenting a system development process in all terms of *software engineering* practices and models. Conversely, we intend to discuss relevant issues concerning the collaborative web browsing and the challenges faced by their developers, presenting justifications about our choices.

1.3 Approach

Along this work, we have adopted a pragmatic approach, organizing the most relevant issues through each phase of OCEAN's development process, in an effort to facilitate understanding. This approach basically consists of standing a set of characteristics in the collaborative web browsing perspective. More specifically, our approach starts at a higher level of abstraction on the development process concerning the *specification* or *conceptualization* phase. The specification phase deals with *what* is the proposed environment. Thus we proceed to the *design* phase in which the focus is devoted to *how* a distributed architecture is able to support a set of required features. Following, in the *implementation* phase, we concentrate our attention on demonstrating the feasibility of the proposed collaborative web browsing system, developing a prototype used as a proof-of-concept. At last, a performance evaluation is executed taking the implemented prototype as a testbed for experiments. The aim of *evaluation* phase is to quantify some measures of interest. Usually the response time is a key metric, given that a collaborative web browsing is essentially an interactive system.

1.4 Thesis Outline

The core of this work deals with specification, design, implementation and performance evaluation issues as the development process of OCEAN advances. This thesis is structured in five chapters.

In Chapter 2, we present background to the collaborative browsing paradigm, discussing some definitions and classifications. We also compare different collaboration approaches and show related work.

Chapter 3 presents OCEAN, focusing on the general features covered by our specification proposal. Such specification is built upon a well-known collaboration theory, the 3C model (17)(18). Once informally specified, OCEAN is formally conceptualized using a more expressive collaboration theory, the Collaboration Ontology(2)(19).

Having formalized all the proposed features, Chapter 4 presents the OCEAN main design issues, regarding how its features could be implemented. The first aspect in this chapter is to define the distributed architecture of our environment, discussing commonly adopted approaches and presenting the components of OCEAN's architecture. Subsequently, this chapter details the protocol proposed for communication among the distributed entities of the architecture.

Finally in Chapter 5, a proof-of-concept prototype is depicted, gathering the most relevant contributions of the previous chapters. This implemented prototype is also used as a testbed for a performance evaluation study, presented in Chapter 6. Such study evaluated the proposal regarding its design with respect to different evaluation perspectives, including experimentations and analytical models.

2 Background and Related Work

The main point on collaborative browsing paradigm is allowing people to collaborate in a traditionally solo browsing activity. This chapter is focused on summarizing the basic concepts behind collaboration, in an effort to reach a better understanding of proper collaborative browsing paradigm. Thus, Section 2.1 briefly describes the research area concerning collaboration from a computational perspective. After that, Section 2.2 explores the collaborative browsing, detailing intrinsic properties of this collaboration paradigm. Section 2.3 presents some related work. Finally, we conclude this chapter in Section 2.4, positioning our proposal in accordance with collaborative browsing specified classifications and some related work.

2.1 Computer-Supported Cooperative Work

The constant growth and evolution of the Internet favors the emergence of new services and interaction paradigms. Amongst them, there are the collaborative one, that have introduced innovative interaction ways among users, providing better results of the collaborative work, either for personal or professional purposes. Such services, also called *groupwares*, are the focus of a great research area, denominated CSCW (Computer-Supported Cooperative Work) (1). In other words, while CSCW includes the universal scientific research field, groupware deals with the respective practical system solutions of collaborative work (20). A classical and well-accepted definition of groupwares was proposed by Ellis, Gibbs & Rein(17):

> Groupwares are computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.

Someone could misinterpret groupware definitions, saying that "groupware" would be just a fancy name for multiuser systems. However, as opposed to common multiuser systems, such as distributed database systems or time-sharing operating systems, groupware systems send a notification whenever something is altered (20). This facilitates users' awareness of each other's existence and concurrent actions. The notification informs all users involved in a group session of modifications in their shared environment. If we take a look at conventional multiuser systems with respect to these notifications, the difference becomes obvious. If a user executes a certain task, such as inserting a new record into a distributed database system or generating a new process in a time-sharing operating system, then other users will not be informed. Instead they must initiate an explicit system query (database query or process listing) in order to be informed of the aforementioned activities (20). Therefore, groupwares are multiuser systems too, but more than that are designed for supporting teamwork, which mainly includes maintaining team's members aware of the supported activities.

2.1.1 Common Groupware Classification Models

We know that groupwares are multiuser system (software or hardware) that support collaboration. However, there are many different ways to collaborate. People usually collaborate exchanging information, dividing work effort or even sharing tools and experiences. So, in order to classify these groupwares concerning collaboration perspective, some works have proposed classification models. The most commonly used are *taxonomies*, which are ways of dividing groupwares into sets (or classes), based on some properties of these systems. These taxonomies are useful, for instance, on discerning the variety of existent groupwares and on identifying similarities among some of them. Two wide referred taxonomies are the *Time/Space*(17) and the *Application Level*(17). Another groupwares' taxonomies can be found in (20)(21).

The time/space taxonomy classifies groupwares into four basic categories, considering the interaction mechanisms provided by the system. These categories are defined by combining two dimensions, *time* and *space*, depicted in Table 2.1.

Table 2.1: Time/Space Taxonomy: the four basic groupware classes originated on combining the *time* and *space* classification dimensions (17) (21)

.

		TIME	
		Same Moment	***Different Moments***
SPACE	***Same Place***	`synchronous local interactions (face-to-face)`	`asynchronous local interactions`
	Different Places	`synchronous distributed interactions`	`asynchronous distributed interactions`

The *time* dimension regards the synchronization of users' interactions. When users are interacting at the "same" time, such as in a meeting or a phone call, these interactions are denominated *synchronous*. When these interactions happen over different periods of time, they are denominated *asynchronous*. Groupwares are thus classified by means of their interactions types. Therefore an e-mail system is usually classified as an *asynchronous* groupware, while a chat tool as *synchronous*. The *space* dimension reflects users closeness requirements for collaborating through the groupware. For example, a collaborative presentation room (22) demands users to be in the same place, whereas an audio conference tool usually does not.

Over the years, a number of different groupware systems were developed to support a specific work situation or a specific range of situations. The range of groupware systems available nowadays reflects the diversity of cooperative work tasks, duration, group size, group location and organizational context (21). Such diversity of groupware characteristics is the basis of the *application level taxonomy*. This taxonomy classifies groupware according to their most relevant characteristics. Due to that, it does not provide a fixed number of categories. Even so, in spite of being less accurate, this taxonomy is quite useful for grouping similar collaboration systems, for instance, in accordance to their goals and what team work support they offer.

Some application level classes frequently found in the literature are described in the sequel. Similar to Farias(21), we do not intent to be complete neither in terms of the classes of systems covered nor in terms of the representativeness of each class of applications. Rather, we aim at giving the reader a rough idea of the diversity of the existing groupware applications and the functionality provided by them for its users. For a more comprehensive description of groupware systems and their application classes, we refer to (21)(17) and (20).

Message Systems: Textual messages can be exchanged asynchronously between team members. Modern systems can handle graphics, images, and even sound and video. The message management is facilitated by additional structural information, such as field for "topic" or "group" (20). Some examples of these message systems are: *e-mail*, *instant messaging* and *audio/video calls*.

Group Editors: also called co-authoring systems or multi-user editors, are used to improve the efficiency and quality of group writing as well as to support the cooperation between authors collaborating during the development of a document. Most of the features frequently found in group editors are related to concurrency control and data sharing, multi-user interfaces, auxiliary communication channels, information storage and retrieval, and the provision of awareness and notification services (21).

Group Decision Support Systems (GDSS): provide computer-based facilities for exploring unstructured problems in a group setting. The goal is to improve the productivity of decision-making meetings, either by speeding up decision-making process or by improving the quality of resulting decisions (17).

Coordination/Workflow Systems: coordination problems mainly arise with asynchronous activities (20), and can be seen as the integration and harmonious adjustment of individual work effort towards the accomplishment of a larger goal (17). To coordinate the activities of team members necessary to archive the common goal, there are four types of coordination systems, depending on the information to be modeled: (*i*) *form-oriented systems*, models the data flow within an organization; (*ii*) *procedure-oriented systems*, models functions and procedures within an organization, like the phases of a formal development process; (*iii*) *conversation-oriented systems*, models the interactions between team members and the resulting actions; and finally (*iv*) *communication structure-oriented systems*, that models complex communication structures within an organization (20). More details about coordination systems, and these four types can be found in (20) and (17).

Conference Systems: are among the most popular groupware systems in use nowadays. Conferencing systems can be roughly split into two categories, computer conferencing and multimedia conferencing. Computer conferencing systems are a variant of electronic mail systems, which allows one to send messages to a uniquely identified place dedicated to the discussion of a particular subject. Messages posted there can be then retrieved and responded asynchronously, by any participant. Whereas multimedia conferencing systems provide at least real-time (synchronous) audio and video conferencing support for remotely distributed participants (21).

Web Conferencing: some multimedia conferencing systems have also integrated support for the exchange of other types of media, such as messages and still images, and also provide a shared workspace (21). Recently proposals in this field have been built using Web technologies, gathering many common web tools into one unique collaboration environment. Such environments have integrated a lot of other minor groupware aiming to provide a complete solution for collaboration through the web, for instance, (*i*) co-authoring of documents, schedules and workflows; (*ii*) presentation and discussion of contents using web meetings, web seminars and audio/video conferences; and also (*iii*) manageable and flexible shared spaces for storing and controlling the collaborative production of this group. Examples of such environments are Cisco WebEx (23) and Adobe Acrobat Connect Pro (24).

2.1.2 The 3C-model

Instead of simply organizing groupwares into sets, the 3C-model(17) appears as a complementary approach for classifying these systems according to the intensity of supported collaboration within a team (20). Despite presented taxonomies, which classifies systems considering their properties, the 3C-model is based on intrinsic characteristics of the proper collaboration concept, being somehow independent of implementation issues.

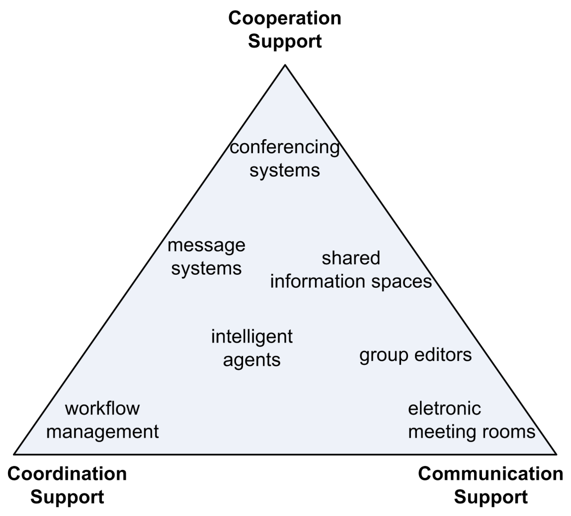
Generally, groupwares are considered "C-oriented" applications (20). This is due to even varying their goals, interaction mechanisms, potential users and collaboration intensity, it is usually possible to distinguish between *coordination*, *cooperation* and *communication*, the three "*C's*". Indeed, these three concepts are the collaboration intrinsic characteristics that compose the 3C-model.

It is difficult to imagine a collaboration scenario without any communication. This way we can see the communication aspect as the basis to any collaborative system. Communication is related to the exchange of information among people (18) focusing on their mutual understanding (20). This can be seen, for example, in conversations between friends, negotiating decisions during meetings or publishing some news to a group. To transmit content, the sender expresses his intentions or goals, defined by symbols in a language that must be understood by all receivers. Moreover, information transmission needs to be accomplished by a communication media (2).

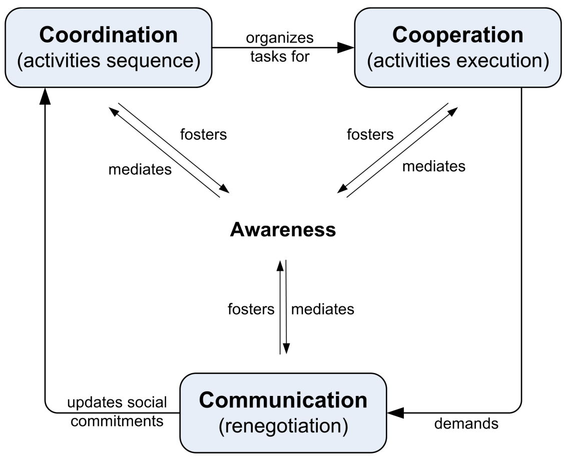
Coordination is related to the management of people, their activities and resources (18). In other words, it aims at finding the best way in which to arrange task-oriented activities and the allocation of resources in the best possible order (20). According to Farias(21), typical examples of coordination problems are the identification of goals, the mapping of goals to activities, the ordering of activities, the selection of actors to perform an activity, the management of interdependencies between activities and the allocation of resources for an activity.

Finally, cooperation is a joint effort in a shared space to achieve some goal (2), being represented by the production taking place in such space (18). In other words, the cooperation aspect is related to the resources dealt with the collaborative work, demanding communication for handling through different people and coordination for organizing how such resources should or could be handled.

Figure 2.1 shows common usages of the 3C-model on classifying groupwares in accordance the three collaboration aspects. One first approach (Figure 2.1(a)) is focused on classifying different groupwares, in the same way that the presented taxonomies do. This type of



(a) Different Groupwares (20)


Coordination (activities sequence)
organizes tasks for
Cooperation (activities execution)
fosters
mediates
fosters
mediates
Awareness
fosters
mediates
updates social commitments
Communication (renegotiation)
demands

(b) Groupware Internal Characteristics (25)

Figure 2.1: 3C-model Usages: common usages of the 3C-model on classifying groupwares.

classification does not organize groupware into specific well-defined classes, however distinguish them by an intuitive measurement about the intensity with which these systems support each one in terms of the collaboration aspects. Thus, they are classified through a similarity on how they support collaborative work.

The second common usage of the 3C-model is on classifying the internal characteristics of a groupware. In this matter, Figure 2.1(b) presents the internal aspects presented in a arbitrary adaptive workflow management system. Especially, this figure also introduces the *awareness* concept as a relevant concept in the model. In short, the concept awareness concerns to groupware users being aware of other users intentions, actions and resources. More important, awareness in collaboration sessions is important for providing coordination and promoting usability (26). Much of the effort faced by groupware developers, especially in synchronous groupware, is related to the provision of awareness. Group collaboration relies not only on explicit communication among the members of the group, but also on implicit availability of information of each other's presence and actions. In this way, awareness can be defined as the ability of the application to expose the activities of the people engaged in a common task (21).

As a remark, the 3C-model was originally proposed by Ellis, Gibbs & Rein(17), however in the same way that Fuks et al.(25)(18) and (2), we have adopted some terminology differences in this work. We consider the term "cooperation" representing what Ellis, Gibbs & Rein(17) denominate "collaboration".

2.2 The Co-Browsing Paradigm

Web browsing is traditionally an individual activity, where a person uses a web browser (e.g. Mozilla Firefox, Microsoft Internet Explorer) for accessing published hypertext documents. However, this browsing can actually be seen as a social event (8), where users share their browsing activity (11).

In this scenario raises the collaborative web browsing paradigm, also known as collaborative browsing (co-browsing) or collaborative navigation (co-navigation). In this browsing paradigm, besides retrieving web documents, a person is able to recommend browsed contents to other people, sharing their preferences or intentions, so making web browsing a collaborative activity. In fact, considering co-browsing a sort of recommendation paradigm figures out a great advantage of this paradigm, that is its applicability on many collaboration scenarios through the web. Illustrating such scenarios, we can see co-browsing on promoting products or contents by their popularity (13), or even in guided content visiting (1) like web seminars activities (24). These examples are basically composed by web pages recommendations, in other words, collaborative browsing. However these examples also indicate that the co-browsing paradigm is itself a broader concept, that could not be precisely classified by presented models.

Indeed, the way of performing recommendations is the most relevant decision on defining a co-browsing groupware. The chosen recommendation method may define the main goal of this groupware, imposing the most basic constraints or features available to its users. In other words, this recommendation method refines the co-browsing paradigm into an specific scope, thus enabling a more precise classification.

There are distinct ways of performing co-browsing recommendations. The simplest is through a standard communication action, where for instance, a person recommends a URL (Uniform Resource Locator) to a friend or a co-worker in order to ask an opinion. However such approach may impose a great overhead to the team work, especially on recommending many web documents for many people. Avoiding such *ad-hoc* approach, an specialized groupware can support the co-browsing paradigm, regarding some general characteristics described in the following subsections. Moreover, these characteristics are independent of design or implementation issues, being intrinsically related to the co-browsing paradigm itself. Design and implementation issues are discusses on next chapters.

2.2.1 Recommendation Method

We consider two basic modes for transmitting a web browsing recommendation, namely *active* and *passive*. In the *active* mode, a person intentionally sends the recommendation to a known destination, that could be another person or a group. Conversely, in the *passive*, more the recommender is observed while browsing, and afterwards the observer entity automatically generates recommendations for other users.

For example, Amazon.com uses this *passive* recommendation concept for suggesting books to buy according to the behavior of other users having similar interests (8). Another example is the GoogleAds, which dynamically inserts advertisement links in web pages related to the web page matter and also the reader's profile.

2.2.2 Interaction Synchronism

This characteristic regards how synchronous users' interactions are. In other words, if they establish a synchronous collaboration session, in terms of the time/space taxonomy.

Usually, co-browsing systems that are based on synchronous session offer a common method for handling recommendations, where they get synchronized through *browsing following relations*. In this method, when one user browses some web page, whereas all other users that are synchronized with him automatically follow such browsing action, thus moving to the same web page. This following mechanism is commonly implemented by the propagation the URLs (1)(11) or propagating the browsed web page content itself (12).

In particular, browsing following relations are commonly called as *master/slaves* (12)(11)(8), being the user who navigates denominated *master*, while the followers are the *slaves*. However, such terminology has been overused in the literature with so many different meanings. Especially, inside the proper co-browsing domain some works have proposed different meanings for master/slaves. For instance, Gerosa et al.(8) has defined "*master*" as a role with exclusive privileges, whereas Esenther(12) has defined "*master*" just as a momentary position of a user, without any coordination behind. In face of that and in an effort to not confuse the reader, we have decided to not use master/slaves terminology.

Therefore we have set three gradual definitions concerning the synchronous co-browsing recommendation methods: (*i*) *browsing following relations*, the aforementioned automatic browsing propagation method; and two specializations: (*ii*) *rigid presenter-attendees*, the specification of roles on following relations, defining one and only one user with rights to browse by their own, while the others synchronized with him or her are meant to be followers; and (*iii*) *flexible presenter-attendees*[1] , which enables users synchronized in a presenter-attendees method to exchange their roles, so giving chances to all users to contribute to the co-browsing session.

2.2.3 Users' Location Requirements

Co-browsing systems can be designed for supporting specific users proximity (time/space taxonomy) requirements during co-browsing activities. This way, there are the ubiquitous co-browsing systems, which allow distant users to co-browse. In particular, this ubiquitous paradigm is the most easily encountered, for instance, on (11)(6)(27)(28) and (14).

On the opposite, there are the co-located co-browsing systems, which are based on supporting near users, for instance in the same room. Such co-browsing approach is very useful on improving lectures and meetings, providing more ways for participants contribute in the event. In particular, ubiquitous systems can be naturally used in co-located mode, the point is that generally, co-located systems provide specialized features, usually regarding hardware integration. For instance, Amershi & Morris(5) have integrated cellphones as telepointers, while Malcher & Endler(22) have integrated projectors and handhelds.

2.2.4 Co-browsing Purpose

Users can use the co-browsing paradigm for supporting different tasks. Even though, co-browsing tools can be specialized or not. Some of these tools are general enough, allowing users to co-browse through arbitrary domains, not mattering their goal, for instance (1)(29)(6) and (30). Despite that, some system are dedicated on performing specific co-browsing tasks, for instance, collaboratively presenting slides (22) or searching content on the web (5). Additionally, another sort of specific purpose is about embedding co-browsing features in specific web applications, which is the approach adopted by (12)(10) and (31).

2.2.5 Coupling Level

Despite the aforementioned interaction synchronism concept which is based on time relations, this concept regards how tight users collaborate. In other words, the coupling level refers

[1]The flexible presenter-attendees method has been proposed by Gerosa et al.(8) with the name of *symmetric* co-navigation.

how much awareness and collaboration facilities are available in the co-browsing session, so indicating the closeness experienced by users, even in a distributed (ubiquitous) scenario.

At the lowest (or loosest) levels of co-browsing coupling, there are systems which only provide, for instance, URL sharing, without any kind of users tight relations (e.g. (6)). At higher levels, systems which provide browsing synchronization methods of shared web pages (*relaxed-WYSIWIS*). Examples of these systems are (1)(11). On even higher coupling levels, specialized features permit users to get even more close (*strict-WYSIWIS*). On such environments users could, for instance, focus on the same content parts (*co-scrolling*, *highlighting*), fill forms (*co-filling*) and update shared content (*co-editing*, *annotations*). More details on coupling levels can be found on (21).

In particular, WYSIWIS is an acronym for "*What You See Is What I See*", that aims at a consistent presentation of shared information to all participants. In its most strict form, WYSIWIS means that all participants have exactly the same context. Thus the screens of all session participants display the identical information (20). Whereas on relaxed forms, session participants could be, in a given moment, browsed to the same web page, but each of them could be reading some totally different piece of this same content.

2.3 Related Work

Many works have proposed solutions in the collaborative browsing area, and some of them are quite relevant to this work. These solutions are grouped through an application level classification are listed below. We do not intend to cover all possible (or existent) co-browsing applications with these classes, they are just an effort for reinforcing relevant systems similarities.

2.3.1 Synchronous Guided Co-browsing

This class consists of groups of users synchronously and actively co-browsing arbitrary web contents using browsing following relations. Amongst the advantages of such applications, the most relevant is allowing distributed users to collaborate through the web, being organized in virtual presentations, also known as web seminars. Especially, this application class definition is usually referred as the proper definition of the co-browsing paradigm, as in (8).

- LiCoB (1) – This is the precursor of OCEAN, and most of facilities proposed by LiCoB persist in this work. LiCoB aims at integrating important features of a lightweight distributed architecture, awareness, session state sharing and annotations (e.g. draw and comment elements over the shared web content). Also, the approach relies on a collaboration ontology that provides a well defined conceptualization and a common vocabulary. Regarding co-browsing characteristics, LiCoB supports ubiquitous and synchronous co-browsing paradigm with active recommendation method, getting synchronized in a flexible presenter-attendees way.

- CoLab (11)(30) – This work proposes CoLab, a new paradigm and tool for collaboratively browsing the web, and it is the most similar to our work. CoLab also supports the synchronous co-browsing paradigm with active recommendation method. Also, it is prepared for allowing its users to co-browse arbitrary web contents being wherever they want.

  This work mainly proposes a new browsing paradigm, which provides coordination flexibility to the collaboration session. In other words, CoLab's users are not necessarily forced to co-browse in a presenter-attendees method. Actually it proposes a workgroup based coordination mechanism for turning co-browsing synchronization even more flexible. Such mechanism allows the configuration of different collaboration scenarios, from rigid presenter-attendees browsing to completely free individual browsing. Another important contribution is that it supports the synchronization of continuous media embedded into co-browsed web pages (*e.g.*, when a user pauses a video, the other users who are following him will also have their video paused).

  CoLab's major shortcomings are the great overhead it imposes on renegotiating workgroups coordination state, and missing of collaboration artifacts when using flexible co-browsing sessions. Exchanging the privilege on conducting the browsing of a workgroup illustrates the coordination problem, whereas revisiting previously co-browsed web pages and collaborate through different workgroups are examples of the artifacts problem.

- IMMEX Collaborative (8) – The IMMEX is a collaboration environment that also has co-browsing capabilities. Such co-browsing module provides synchronous sessions with a common presenter-attendees organization. In this case, this system imposes coordination rules where only one user has browsing rights (the presenter), while the others are only allowed to observe (or to "follow") his/her actions.

  In addition to that, IMMEX has improved its coordination mechanism using a flexible presenter-attendees based on a token-passing-based mechanism, thus enabling users to change their roles. The authors denominate this mechanism as *symmetric* co-browsing, since all users have chances to be masters, presenting whatever they want. However, any user can decide to take the token, no token retention mechanism is provided (11).

- PageShare (31) – A commercial tool for providing co-browsing capabilities. Basically provides basic communication tools, such as simple annotations and collaborative objects manipulation, as filling forms.

- Browzmi (14) – Social network browsing recommendations is the focus of Browzmi. The major advantage of this system is that it allows users to co-browse though different paradigms. It could be active or passive, and synchronous or asynchronous. In other words, the users can join the same session at the same time, following a presenter-attendees approach, or not.

  During a Browzmi session, being alone or not, users are able to browse and recommend favorite content, by using Browzmi collaboration tools. For instance, this system offers (i) *clipping*, for recommending specific pictures or other embedded medias; (ii) *rating*, for classifying the recommendation; and (iii) *personal comments*, for publishing a statement.

  Such recommendations feed participant's profile on Browzmi social network. In doing so, even that a person uses this system alone in a session, he or she can still co-browse with other people, based on the recommendations made all over the session.

2.3.2 Domain Embedded Co-browsing

Co-browsing is a widely applicable collaboration paradigm. However some multi-user systems may offer co-browsing features for supporting some specific activities. For instance, helpdesk portals allowing their users co-browsing an enterprise web application, in a tutoring session. The most important is that systems like this helpdesk portal, may need a co-browsing feature specialized to their needs. So, general purpose co-browsing tools usually not achieve all these applications' requirements.

In this matter, domain embedded co-browsing gathers systems developed with the intention to offer co-browsing capabilities to specific domains. It means that they usually need to be somehow integrated to the target systems, for example, in a source code level integration or through application programming interfaces (API).

- CWB - Collaborative Web Browsing (12) – This proposal is focused on the basic features for providing synchronous and active co-browsing. The great advantage of this proposal is that it replicates whatever one user does for the others. These interactions can be, selecting a piece of text, filling HTML forms and even where the mouse pointer is in the web page.

However, CWB has many disadvantages. For instance, it is restricted to an specific domain (*e.g.*: `*.ufes.com/`), and has to be installed in the same domain that a web service which is desired to co-browse. Besides, it does not provide any coordination mechanism for organizing the browsing following relation. In other words, CWB does not prevent two different users browsing different web pages, at the same time. In this case, users cannot forecast what could happen, since CWB will accept the browsing action which firstly arrives in the server. Thus, the slowest browsing action will be just dropped, and how more users make simultaneous actions more confusing the session would become. Another restriction concerns its user interface, which is quite confusing, not making users aware of what is happening in the session, and moreover, who is doing what in the session.

- Clavardon (10) – This is a commercial tool specialized on embedding co-browsing synchronous sessions into web services, in particular, e-commerce applications. Besides, Clavardon also provides an online co-browsing service for browsing through arbitrary domains, using a rigid presenter-attendees approach.

  In addition, the most relevant features on Clavardon is the ability of highlighting content parts, synchronizing participants scrolls, and also, allowing users to jointly filling forms. This way, Clavardon provides a higher coupling level towards a strict-WYSIWIS collaboration session.

2.3.3 Co-located Environments

As aforementioned, co-located co-browsing systems are designed for supporting participants in a closeness scenario, for instance inside the same classroom. Systems like these are usually augmented with integration to this shared physical environment, in order to take advantage of this participants proximity, and thus fostering a richer collaboration experience.

- CoSearch (5) – This work proposes a system for collaboratively search content on the Internet. However, its particularity is that it was designed to support users gathered around a single computer to work together in collaborative searching tasks. Such collaboration is supported by offering multiple inputs to the same environment. In this case, users that are not controlling the computer, can also contribute to the group task using his/her cellphone as an additional input mechanism (telepointing).

- iPH (22) – The Interactive Presenter for Handhelds (iPH) is a groupware for supporting presentation of slides in a classroom. Besides having integration with a projector, attendees in this environment can use personal handhelds for contributing to the ongoing presentation, for instance making annotations over presented slides. However, all users in the same session still follows a rigid presenter-attendees synchronization method, where the presenter has the browsing rights, in this case, the privileges for choosing slides.

  Especially, a disadvantage of iPH is its too restrictive system requirements. For instance, it is completely designed for working on Microsoft Windows platform, even in mobile devices.

2.3.4 Loosely-coupled Browsing Recommendations

As defined, people can collaboratively browse independent of browsing following relations, just sharing their browsing recommendations, despite synchronous or asynchronous interactions. Groupwares that support these interactions, usually provide a collaborative production shared space where users can feed with browsing recommendations, and thus sharing with other participants. It characterizes a great cooperation degree, regarding the 3C-model.

In particular, such loosely-coupled browsing recommendation approach is often not considered co-browsing, however its similarities with all mentioned co-browsing characteristics made us to consider these applications as a co-browsing class too.

- GUH - Group Unified Histories(6) – This work also presents a system which allows users to get together in a session and synchronously co-browse the web. Especially, such sessions are augmented with different collaboration tools, allowing users to express their opinions through many different ways, for instance, through a chat room, or rating browsed contents.

  Contrary to common co-browsing approaches, in GUH there is no predefined browsing synchronization relation, like the presenter-attendees synchronization method or at least browsing following relation. Instead, every user is free to browse the web on their own. Even though, this system presents an important cooperation advantage, as users' browsing activities feed GUH's session shared history. From this shared history, every participant can check out all browsing activities of all session members.

  Thus, even not offering any browsing following mechanisms, this system allows users to make browsing recommendations, and moreover, to discuss such recommendations. However, such unconstrained co-browsing approach can damage the quality of collaboration when the session grows up. Coordination mechanisms could prevent lost of information in such scenarios.

- Kiobo (28) – social navigation is based on a totally passive recommendation paradigm (9). This system observes users individual browsing activity, composing a unique repository. All these information combined, can provide social browsing recommendations. For instance, a user can see what subject have been more browsed recently, without necessarily knowing who has browsed each content.

- Blocool (13) – such system passively observes which blogs[2], in particular blog posts, have been read by users. Such observations feed a reading history in the user's profile. These histories are commonly useful for *bloggers* (blog owners), who wish to publish an automatically generated *blogroll*[3], containing his/her recent readings. In other words, Blocool tries to answer a simple question: "What blogs am I reading?".

  In a blog maintained by a Blocool user, readers can see which material this *blogger* is using as reference for his posts. In addition, Blocool also offers an asynchronous loosely-coupled following mechanism, where users pre-define other users to be followed, after that he/she can automatically receive blog recommendations of his friends (followed users' readings) that are more related to his profile.

  Due to its characteristics, Blocool approach can also be classified as collaborative browsing, as blog owners can use Blocool for recommending other blogs to their readers. Moreover, the co-browsing mechanism applied here has characteristics of passivity, asynchrony, ubiquity and presents the specific purpose of recommending blog readings. Another similar but less expressive applications are proposed in (32) and (33)

2.4 Positioning our Proposal

In this work, we focus on a subset of collaborative browsing, where distributed users get together in a synchronous session for actively browsing through shared web content. This subset is also referred as the proper definition of co-browsing (8), thus, in this work, we refer to this subset as *co-browsing*. Considering specific characteristics, OCEAN relies on some related work solutions structured in accordance with the 3C-model (Section 2.1.2):

[2]A blog (a contraction of the term weblog) is a type of web site, usually maintained by an individual with regular entries of commentary, descriptions of events, or other material such as graphics or videos. Entries are commonly displayed in reverse-chronological order. <en.wikipedia.org/wiki/Blog>

[3]A blogroll is a list of links to other blogs or web sites that the author of the blog regularly likes to read. The blogroll generally resides in one of the side columns of the blog. <*The Blogosphere's Dictionary*: www.blogossary.com/define/blogroll>

- Regarding coordination, OCEAN's approach is based on flexible presenter-attendees. Such coordination approach is designed by means of users' roles and privileges which provides simple session management, regarding both system's designer and final users points of view. Similarly to CoLab (11), OCEAN presents even more flexibility with the addendum of independent and synchronized groups, where a large variety of coordination scenarios can be performed, as opposed to over-constrained approaches like in (1) and (8).

- Considering cooperation aspects, OCEAN inherits collaborative production shared space concepts defined by GUH's shared histories (6), Kiobo's repository (28) and Blocool's blogrolls (13). In other words, our proposal takes in consideration a shared space for aggregating the many productions of a co-browsing session, in an effort to maintain the participants tight related even when working independently.

- Finally on communication, OCEAN allows users to make annotations (draws and text notes) over shared web pages. For example, such annotations are useful for expressing thoughts and, getting the focus of attendees on what the presenter wants to highlight. Similar annotations concepts can be found in Clavardon(10), PageShare(31) and CWB(12), however, the way it is implemented in such tools is not expressive enough since users are just allowed to highlight parts of texts.

Recall that the main objective of this work is to propose a general co-browsing system that supports properly the three collaboration aspects. However, we are not interested on attending every distributed workgroups' needs, proposing a general collaboration solution like web conferencing systems.

In the sequence, next chapter specifies and formally conceptualizes OCEAN, describing all features that compose this proposal through the prism of the 3C-model aspects.

3 The Proposal: characterizing main features

The objective of this chapter is twofold: (*i*) presenting the characteristics and features of OCEAN, (*ii*) formalizing conceptual models. In an effort to keep our work in accordance with what has been produced in the CSCW field, we follow the widely adopted 3C Model (17)(20), trying to divide our discussion, specifications and models into three main aspects: cooperation, coordination and communication. This approach allows us to deeply discuss each collaboration aspect, favoring the understanding and validation of the identified concepts (25). Moreover, this distinguished view can support parallel comparisons with other groupwares, in particular another co-browsing systems.

3.1 Proposal General Description

This section presents an informal specification of OCEAN. This specification considers *which* features our co-browsing service offers without showing *how* such features internally work. Detailed design and implementation concerns are discussed on further chapters. In the following, the specification is presented in terms of three distinguished points of view. Each perspective is guided by one of the 3C-aspects, and contains a discussion of the collaborative browsing paradigm intrinsic characteristics, and the OCEAN proposal.

3.1.1 Coordination Aspect

In the context of collaboration sessions, coordination appears to be a major aspect for keeping controlling or management. Commonly, this management is made by imposing constraints in order to control participants' activity. This way, we could suppose that, the more restrictions our groupware impose, the more coordinated it might be, and, accordingly, more collaborative.

The answer for this question can be visualized making a comparison with an example out of the computational context: an arbitrary Brazilian public service. In this case, the service's workers, the client and also the government are the participants of a collaboration session, which has the objective to solve the client's problem (or should have). But this session is coordinated by various rules. For instance, the client should deliver several documents, each of them must be reviewed by a different worker from different sectors on different days. This massive set of coordination rules turns the service bureaucratic, making the collaboration a daunting task. Based on such example, the answer might be "No", because an excess of coordination could "over constrain" users' contributions, turning the collaboration almost impossible. Conversely, not imposing any coordination could also be harmful for the quality of the session. Therefore, the point is to find an equilibrium, allowing users to collaborate in a certain manageable freedom state (34)(35).

Focusing on collaborative web browsing, the coordination aspect is commonly implemented based on offering (or not) browsing synchronization facilities. In other words, such coordination usually has one of two opposite approaches: *unmanaged*, found in (12)(6)(28) and (36); and *presenter-attendees*, adopted in (11)(5)(8)(27) and (22).

In the *unmanaged* approach, users can do whatever they want during the session long. So, co-browsing session participants rely on informal coordination, in which there is no predefined flow of work or privileges hierarchy, and coordination is handled by actions initiated by people themselves on an ad-hoc basis. Such informal coordination is supported mainly by computer-mediated communication systems (34), like conferencing systems. However, this lack of a formal control can generate a participation chaos in the session.

Such chaos problem becomes evident considering synchronization methods (browsing following relations) adopted by many approaches (11)(14)(22), including our proposal. In these cases, when one user browses to a different web page, this browsing action is propagated to all synchronized users, and after that these participants automatically browse to the same web page. Note that, if every participant were allowed to browse, then concurrent browsing actions may occur, generating confusion and damaging the quality of the session.

Proposed by Esenther(12), CWB[1] is an example of a co-browsing system that is based on the *unmanaged* coordination approach and also offers browsing following relations. Due to that, its users can experience confusion on executing concurrent browsing actions. In a contingency effort, when a concurrent browsing action occurs, CWB is designed to propagate just only the action which first reaches its central server, dropping out other concurrent actions. Such solution maintains the co-browsing session in a consistent state, however may generate confusion. For

[1]This system has been discussed in Section 2.3.2.

instance, when two users are synchronized in CWB, and both concurrently browse to different web pages, the user who had unsuccessfully browsed (last reaches the central server) is moved to a web page different that the one he had chosen.

The second commonly adopted coordination model is *presenter-attendees*. At any moment, only one user performs the presenter role, while all the others perform the attendee role (rigid presenter-attendees). A presenter is responsible for guiding the session, being the only one allowed to choose the contents to visit, whereas the attendees are just allowed to see what the presenter has chosen. In other words, the presenter and attendee roles stands for interaction constraints, avoiding concurrent browsing confusion in the session, and thus managing the browsing following relations. Hence, imposing rigid rules to the session, this approach avoids concurrent browsing. Conversely, if such rules are too restrictive, blocking attendees to also contribute for the session may damage the collaboration.

Some works, as the one proposed by Gerosa et al.(8), have evolved the presenter-attendees approach, allowing users to dynamically exchange their roles (flexible presenter-attendees). In other words, the current presenter could assign his/her presentation rights to one of the attendees. This role exchange protocol increases the flexibility of a co-browsing session, giving to users equals opportunities to participate. Even so, such improvement still is too restrictive, since it keeps users locked into tight browsing following relations, restricting their contributions by only one user at a turn.

A co-browsing session should be even more flexible in order to allow its participants to independently contribute to the shared goal. It means that the users should be able to browse documents by their own, without stopping to contribute to the session. It could be useful for a divide-and-conquer strategy, where for instance, the participants want to collaboratively search for some information on the Internet. In such scenario, each participant search through distinct paths, collecting and sharing interesting data with the rest of session members. This strategy is commonly present in co-searching tools (5), but also appears in some co-browsing tools (6). However, the presenter-attendees model still is very useful, whenever a guiding behavior is necessary. For instance, during lectures, online tutoring or any other kind of presentation tasks (1). Hence, an hybrid coordination mechanism enabling these two distinct strategies, divide-and-conquer and presenter-attendees[2] becomes necessary. Such a mechanism should be flexible, in order to allow the participants to migrate between both co-browsing strategies at any time, without affecting the continuity of the collaboration session.

[2]From now on, when mentioning "*presenter-attendees*" we are referring to the "*flexible presenter-attendees*" coordination model, considering presentation roles exchanging mechanisms.

For this reason we propose a coordination mechanism designed to provide flexibility for dynamic co-browsing sessions. This mechanism is divided in two coordination levels. The first deals with subgroups of participants, supporting them to work following the divide-and-conquer strategy. In other words, session members could get organized into subgroups where each of these subgroups is independent to browse. Following, the second coordination level deals with subgroups' internal coordination rules. Internally, each subgroup plays the role of presenter-attendees strategy. So, at any moment, there is only one participant with browsing privileges in the same subgroup, while all the others in the subgroup are attendees. In particular, the presenter role can be assigned to any other subgroup's participant (flexible presenter-attendees), for instance through token passing mechanisms (20).

As a consequence, such division of users into subgroups leads to the formation of collaboration sub-sessions inside the co-browsing session. Note that, when co-browsing, the participants of a subgroup share information and characteristics that are specific of this subgroup, likewise a goal, a common production space, awareness information and etc. In fact, it characterizes that these people (subgroup members) are participating in a proper collaboration session. Therefore, OCEAN coordination mechanism enables the formation of specialized collaboration sub-session allowing groups of users to work on their specific goals, but still participating on a great co-browsing session, for a major shared goal. Additionally, in the scope of this work we have denominated such collaboration sub-sessions as *co-browsing threads*.

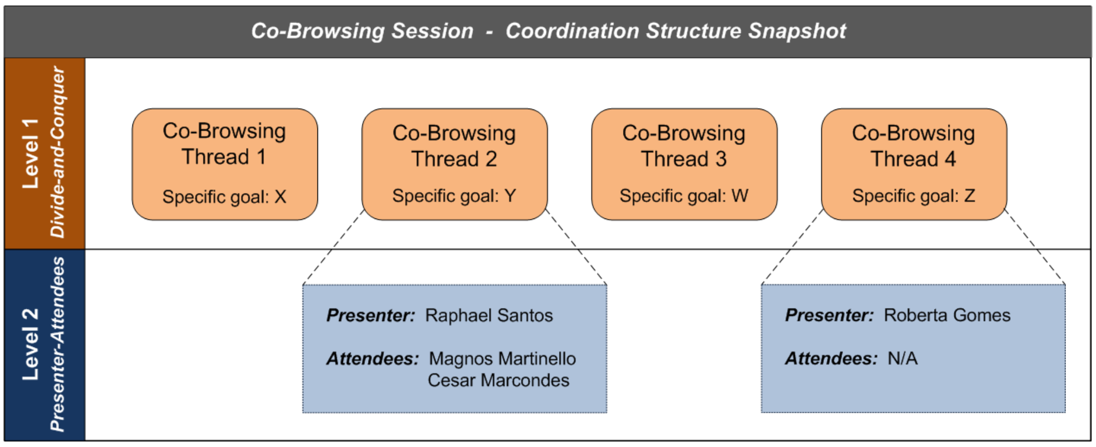


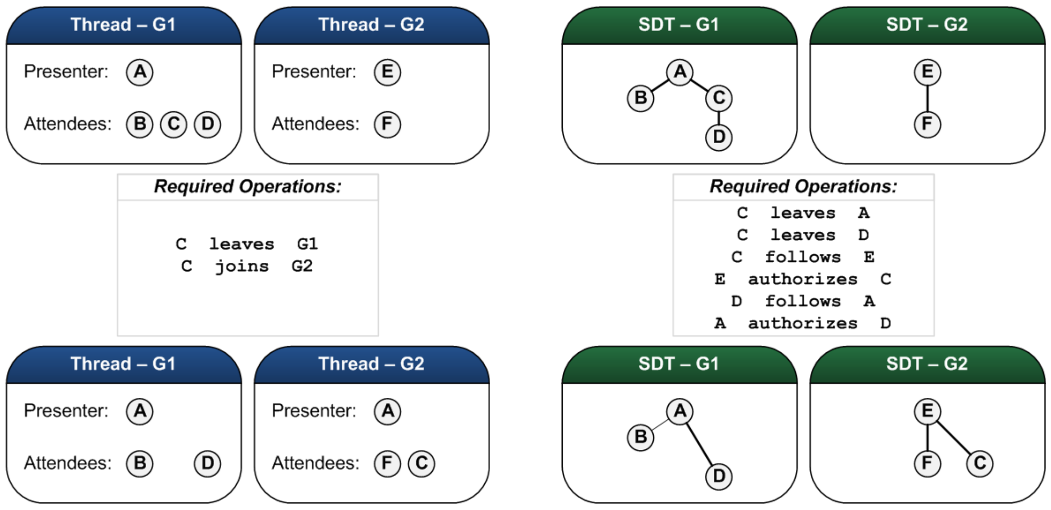
Figure 3.1: Coordination Levels

Figure 3.1 depicts an arbitrary scenario of a collaborative browsing session, focusing on the coordination levels. Regarding Level 1, this session presents four independent co-browsing threads. Regarding Level 2, the internal organization of threads two and four can be noted. Especially, thread four presents an special case, when there is only one participant. This particularity allows users to navigate alone, but still contributing for the co-browsing session as a whole.

As a result, this 2-level coordination mechanism provides a high degree of flexibility to the session, without losing control. It can be noticed that this mechanism also supports the original coordination approaches, *unmanaged* and *presenter-attendees*. For instance, if every session member wants to navigate as in the unmanaged approach, then they just should create individual subgroups (co-browsing threads). But now, this unmanaged use does not cause the aforementioned confusion, since there are no browsing following relations between participants of different threads. On the other side, if they want to stay all together in the same presenter-attendees approach, participating of a web seminar or a virtual lecture, they just need to join the same thread.

Moreover, the OCEAN's coordination mechanism is quite simple to manage. OCEAN provides coordination primitives in order to easily re-configure the session, without overcharging the user with too many commands or workflows for example. The primitives are the specific coordination actions supported by the mechanism, and are associated with the related task level. Considering the groups management, on coordination level 1, the primitives are: *create thread*, *join thread* and *leave thread*. Using these primitives, any user can respectively, create a new co-browsing thread, join an existing co-browsing thread or leave the co-browsing thread he/she is currently participating in. Likewise, the coordination level 2 provides: *privilege grant* and *privilege revoke*, used for managing privileges for users action and information access. For instance, these primitives are used for exchanging the presenter role between two participants, reconfiguring their browsing privileges. More details on coordination primitives usage are presented on chapter 4.

Hoyos-Rivera(30) proposed an elegant coordination mechanisms for a co-browsing system called CoLab, which is also based on session workgroups. This division into workgroups provide coordination flexibility for CoLab. Even so, CoLab's coordination is centered on formal synchronization relations between pair of users. Such relations can be viewed as "browsing following agreements", where one user follow the other to whatever site this last navigates to. The basic restriction in this schema is that it is not possible to simultaneously follow different users. However, a user can be followed by more than one. Besides, while following a user, you can simultaneously be followed by other users. Accordingly, a hierarchical structure can be formed during a session. Such hierarchical organization is denominated SDT (Synchronization Dependency Tree). Whereas the root user is the only one with navigation privileges while the other nodes transitively follow the root user. So, each SDT formed in a CoLab's session remains a workgroup. Despite the great flexibility provided by this model, changes of the coordination scenario can demand to much effort from the participants, on re-establishing following agreements.



(a) OCEAN (b) CoLab

Figure 3.2: Comparison of OCEAN and CoLab coordination mechanisms

Figure 3.2 depicts a coordination scenario example, where the user *C* moves from group *G1* to group *G2*. Looking at OCEAN's approach (Figure 3.2(a)), this operation is made on *C* calling the *join thread* primitive for group *G2*. In fact, the *leave thread* primitive is automatically called by the system, since in OCEAN's coordination model a user cannot participate in more than one co-browsing thread at the same time.

Now regarding the CoLab's approach (Figure 3.2(b)), users A, C and D are directly affected by two "relation dissolve" and two "creation". Particularly, for each synchronization relation establishment it is mandatory an authorization of concerned participants. A remark is that for any relation re-establishment internal to a workgroup structure (for example, in order to carry on a group privileges exchange between two users), the same operational overload will occur.

Hence, OCEAN provides a flexible and simple manageable coordination mechanism. In this model, users can create specific collaboration sub-sessions committed to a specific goal, all inside the same co-browsing session. This way, when a single person leaves this sub-session, this goal commitment can persist. Such behavior is similar to the division into departments of an arbitrary company, where an event in which an employee leaves a department, this department usually maintains its identity and objectives. Conversely, CoLab just supports commitments between users' pairs, not allowing the attendees to explicitly indicate interest in following a certain shared goal in the session.

Nevertheless, having such user based coordination in a co-browsing session would be interesting to complement the OCEAN model, increasing thus its flexibility and expressiviness. However, we decided not to handle this at this moment, since just porting CoLab's model to our system would not be a priority contribution. In the future we intend to study how such user based coordination could contribute to OCEAN's co-browsing sessions.

3.1.2 Cooperation Aspect

According to the 3C-model, cooperation is the production resulting from the collaboration activity that occurs inside shared workspace (25)(18), either for real or virtual spaces. Taking the Google Docs[3] as example, the cooperation is mainly represented by the documents that are produced in such system. Considering that Google Docs also offers additional communication features, the messages exchanged in the sessions are also part of the cooperation. Actually, the cooperation can be materialized by the set of collaboration artifacts produced in a collaboration session. In addition to that, according to Nguyen, Rekik & Gillet(37), the purpose of the concept of collaboration artifact is to serve as a bridge that connects agents and software, providing a shared workspace for the participants.

So, what means "to cooperate" within collaborative web browsing? Remaining to previously adopted definition, collaborative browsing aims at allowing groups of users to actively share their browsing activity, recommending web contents and following received recommendations. In face of that, cooperation in the collaborative web browsing paradigm lies on the participants' browsing activity. Thus, the main artifacts produced in a co-browsing session are the web contents browsed by the group, where the cooperation can be materialized through a Co-Browsing History.

Standard browsers usually create browsing histories only recording the visited URLs. This approach is not enough to express cooperation in a co-browsing session. In fact, other relevant information is produced during each content visualization, such as, the established coordination agreements. Particularly, it is very important to know who was the presenter that has chosen each browsed content. This authorship awareness promotes user ideas on contextualization, and can be applied to measure the relevance of the created information based on its source (38). Moreover, messages exchanged through provided communication features should also compose this co-browsing history, since they are part of cooperation too.

Therefore, the co-browsing history proposed here is a registry that stores all events occurred in a co-browsing session. Such registry is a structure composed of minor independent registries,

[3]Google Docs, an online collaborative editor for documents, spreadsheets, forms and presentations. Available at: <http://docs.google.com> Release: 02/2009

where each of them concerns a co-browsing thread in the co-browsing session, being composed by the events occurred strictly in this sub-session. These events could be users' participations (*e.g.*: co-browsing action) or even system's management actions (*e.g.*: determining a participant logout because of some network communication unavailability).

Especially, browsing participations (the act of choosing and loading a URL) are classified as *checkpoint* events. This is due to the fact that each co-browsing action takes the group to a different collaboration context, possibly changing the group discussion focus. All other events occur over one of these contexts. We understand that a simple change of URL could sometimes not change the collaboration context of a group, for example, when this URL access concerns a page changing of a same document. In this matter, we intend in the future to make this checkpoint definition flexible, allowing for instance that users or an specialized agent could dynamically determine which events are checkpoints and which are not.

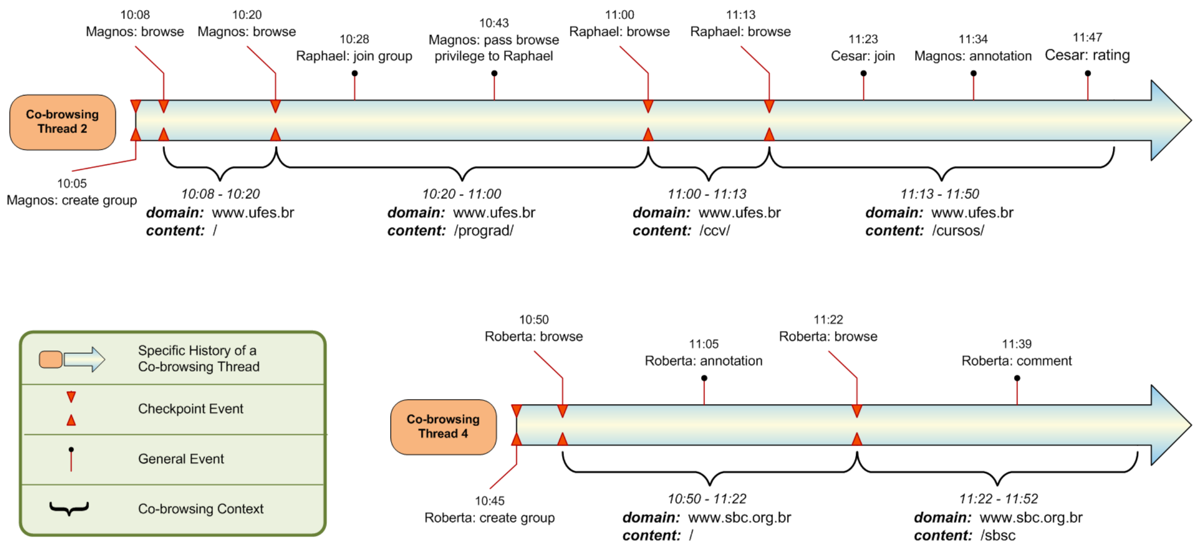


Figure 3.3: Co-Browsing History Example: a schedule diagram representing the co-browsing history of an example co-browsing session.

Following the example depicted on Figure 3.1, which was focused on coordination, Figure 3.3 shows its cooperation perspective, presenting a co-browsing history. In this case, only two threads are depicted, containing events and also the associated awareness information, namely: authorship, context and occurrence time.

Besides that, the proposed co-browsing history may motivate and increase collaboration, since it provides extra control information for supporting co-browsing threads, if compared, for example, with CoLab (11). In the CoLab's coordination model, users are able to make independent groups and so on. However, CoLab does not provide any cooperation advantage for using such division strategy, since its subgroup (SDT) does not explicitly share any specific collaboration artifact (such as a subgroup history).

In CoLab, if a user wants to independently browse, he would need to leave the current SDT, browse to wherever he wants, and after that re-join that same SDT, in order to share his experiences. But, such specific sharing activity is not supported by CoLab. So this user faced the overhead of leaving and re-joining an SDT without any advantage of using CoLab browsing interface. It would be easier for this user, to just open a new browser window, using traditional single browsing paradigm, and concurrently, commenting his experiences in the SDT's group. This shortcoming is due to CoLab's SDT works as completely disjoint sessions, not sharing any information.

Conversely, our approach is based on sub-session which shares every produced artifact with the entire co-browsing session containing this sub-session, since all events feed the same registry structure. In the same way as Oliveira, Antunes & Guizzardi(2), we consider a collaboration session representing a period of time in which participants are engaged to collaborate with each other for a common purpose (2). Note that OCEAN allows users to keep cooperating, even when they are working individually (in different threads), since they keep contributing to the session's browsing history. This motivates users to keep using the system, even for performing independent tasks that could be helpful to the co-browsing session as a whole.

3.1.3 Communication Aspect

According to Fuks et al.(25), the communication aspect is related to messages exchanging and negotiations by participants. In collaboration sessions, senders transmit messages which expresses their opinions, intentions and goals, in order to exchange knowledge with other participants (2). Besides, shared communication channels are often used to coordinate interaction with other collaborative functionalities (39). In particular, communication can be used to negotiate common goals or to solve conflicts. For instance, participants usually negotiate coordination scenario changes through communication (34).

Regarding co-browsing scope, the collaboration essentially lies on the web content recommendations. Communication can be really relevant for supporting this co-browsing activity, for example for promoting knowledge dissemination or for providing means of negotiating content relevance and leadership privileges during presentations. This way, general purpose communication features like the ones supported by chat and audio/videoconference applications are interesting solutions to be used during co-browsing sessions, since they enable users to freely communicate.

In fact, in order to work collaboratively, people need to communicate (40), and moreover, good communication encourages collaboration. However, considering some co-browsing scenario, just offering a general communication feature, like a chat, can cause overheads during conversations and loss of communication efforts. Imagine a virtual lecture containing large web pages presentation though a co-browsing system. In such a scenario, the presenter usually needs to call the attendees' attention for an specific content. In this case, if the only available communication tool were a standard chat room, then the presenter would need to write down all the directions to inform attendees about the content piece localization, before starting the real relevant discussion. By the way, this problem also happens for other general communication medias, such as audio conferences. This overhead is due to the fact these general purpose communication features are not tightly related to co-browsing contexts and artifacts.

As the focus of OCEAN is providing efficient co-browsing mechanisms, providing general communication functionalities might not contribute to the system's specification. On the other hand, general communication tools are still valuable for free collaboration, and should not be ignored. Accordingly, instead of defining them as part of OCEAN's specification and re-implementing them, a good strategy for using such features is to provide means for integrating existent communication tools to the co-browsing system. Such strategy was adopted by Lima et al.(41), showing the integration of a co-browser with an audio conference system.

However, systems integration is not a simple nor a trivial task, since there are many issues to take in consideration. One example is the integration level, that is about how coupled the integrated systems could and should be, considering for instance integration of user interfaces, stored data or happening events. Another important issue to consider is the adopted integration method. Some commonly adopted methods are: (*i*) the ad-hoc approach, based on applications' source code modification; (*ii*) using API (application programming interface); or even (*iii*) relying on integration frameworks, like OpenSocial(42) or LEICA(43). We indicate as further research the investigation of how interesting is for applications to be integrated with OCEAN, and also, how such integration should be conceived.

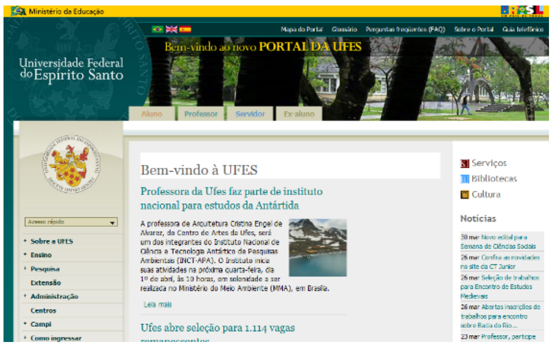
While it is interesting to support general communication functionalities through external tools, considering OCEAN's built-in features, the specification of specialized communication features could improve the communication quality, favoring collaboration effectiveness in the co-browsing context. Therefore, as we are focused on co-browsing paradigm specific problems, we specify the communication aspect only considering a specialized view. Considering that the most relevant resources handled in collaborative browsing are the shared web pages, specific communication features should be aware of them. Some co-browsing systems already offer specialized communication features, such as: content commenting (14)(6), text highlighting (31)(10)(44), objects clipping (14) and voting or rating dissemination(6)(33).

All the aforementioned specialized communication features are very helpful for improving the usability of any co-browsing system. However, only re-implementing such features in our proposal would not be a relevant contribution. Besides, these features may be too specific, being useful only for some scenarios. For example, a voting feature will probably not be useful in a lecture scenario, where the presenter is just passing information to the attendees. In order to chose a communication feature, we tried to find a good relationship between generality and specificity, as to support communications that could be, somehow, (i) explicitly associated to the collaboration artifacts, and (ii) useful for almost any co-browsing scenario.

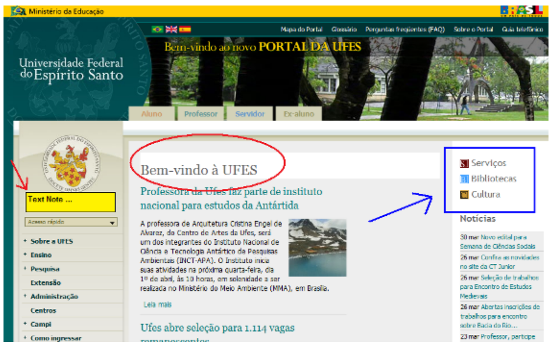
The choice was for the Annotation feature, a subset of the digital ink concept(45), to mainly represent the communication aspect of our proposal. Generally, an annotation feature stands for giving to a user the ability to make notes and marks over a visualized content, with the intention of pointing some specific part of this content, so calling other viewers' attention. Such facility is quite useful on reviewing tasks, for instance. It can be seen as a person reviewing a printed text, making notes with a pen. This feature is usually encountered on collaborative document manipulation systems, for instance, Microsoft Word, Microsoft Visio and Adobe Acrobat. Contrary to traditional browsing paradigm, where users are isolated document readers, in the collaborative browsing, they are just sharing the act of reading these documents. In such scenario, an annotation feature provides the ability to dynamically review the shared document. For example, while accessing online lecture materials, such annotations can help students to focus on the key points of a lecture. Also, it can enable students to engage in a discussion (46).

Annotation feature is supported in Clavardon(10), PageShare(31) and WebAn(44). This is an important resource, allowing the participant to share information. However, the way it is implemented in such tools is not expressive enough since users are only allowed to highlight parts of texts. In OCEAN, annotations are allowed over the entire shared content, including images and other embedded media (1). This feature makes the collaborative browsing more powerful as it reduces the need for additional collaboration tools, enabling participants to share contextualized comments on each other's content. Besides, annotations help preventing information loss, thus being very important to track and reproduce the collaboration session.

OCEAN's annotation feature comprises the ability to make geometric draws and text notes, similar to a white-board system, however these strokes are painted over the shared web page. Additionally, the produced annotations are tightly related with a web page, as an ink mark. So, when a presenter moves to another web page, this new page appears clean. The annotations are kept in the co-browsing context, where they were placed. For instance, Figure 3.4 shows an example of an annotated web page (www.ufes.br). More details about supported stroke types and implementation issues are presented on next chapters.


Universidade Federal do Espírito Santo
Bem-vindo ao novo PORTAL DA UFES
Bem-vindo à UFES
Professora da Ufes faz parte de instituto nacional para estudos da Antártida
Ufes abre seleção para 1.114 vagas
Serviços
Bibliotecas
Cultura
Notícias

(a) Original Web Page


Universidade Federal do Espírito Santo
Bem-vindo ao novo PORTAL DA UFES
Text Note ...
Bem-vindo à UFES
Professora da Ufes faz parte de instituto nacional para estudos da Antártida
Ufes abre seleção para 1.114 vagas
Serviços
Bibliotecas
Cultura
Notícias

(b) Annotated Web page

Figure 3.4: Annotations Example: in this example, a user has painted an ellipse, a rectangle, two arrows and have written one text note.

Another important aspect to consider in the communication scope, is supporting negotiation. Actually, negotiation can be viewed as a specific case of communication in collaborative systems, mainly related with the communication process necessary to take decisions in group. Considering an ordinary collaborative browsing session, coordination decisions could require some negotiation, either for redefining groups or for internal groups policy changes. Thus, we propose two communication primitives in order to facilitate coordination negotiation. They are:

- *Join Invitation* - through this communication primitive, users participating in a co-browsing thread T can invite other users to also become members of T. The invited users just receive this message, but he or she is not obligated to formally reject or accept such invitation. The group join invitation just works as predefined message or suggestion. Its receiver can just ignore it, in case of not considering such invitation interesting. So, this communication primitive can facilitate threads constitutions negotiation.

- *Privilege Request* - according to the OCEAN coordination mechanism, inside a group, the current presenter could give the *Presentation Privilege* to any other group member at any time, characterizing the *flexible presenter-attendees* synchronization method. The privilege request communication primitive, allows the attendees to expose to the group, especially to the current presenter, his/her intention to be a presenter. In the same way, the current presenter can accept such request or not. Moreover, the acceptance has not to be at the right request moment. The current presenter becomes aware of the requester intention, and he/she can give the privilege at the moment he/she considers more adequate.

Hence, OCEAN provides communication facilities specialized on characteristics of the collaborative browsing paradigm, but still is general enough, that are useful in almost any co-browsing scenario. Also, these facilities are inter-related with the other two collaboration aspects. Considering coordination, the negotiation primitives provide means of discussing and suggesting new session arrangement scenarios. Whereas, the annotations feature permits enhancing the cooperation, aggregating value to the session's shared production space.

3.2 Conceptual Formalization

In the previous section, OCEAN was informally specified, distinguishing its main characteristics on three collaboration views. Aiming at a correct and consistent development of our co-browsing system proposal, this section evolves previous specification, formalizing relevant[4] concepts and relations.

At first, in order to summarize the informal specification into a single view, Figure 3.5 depicts the inter-relations among the described 3C aspects (25). Figure 3.5 shows that, all the aspects are inter-related with the *Awareness* concept, which stands for collaborative state of consciousness of session participants and is relevant, for instance, for promoting groupwares usability (26). Additionally, this figure is a specialization of Figure 2.1(b).

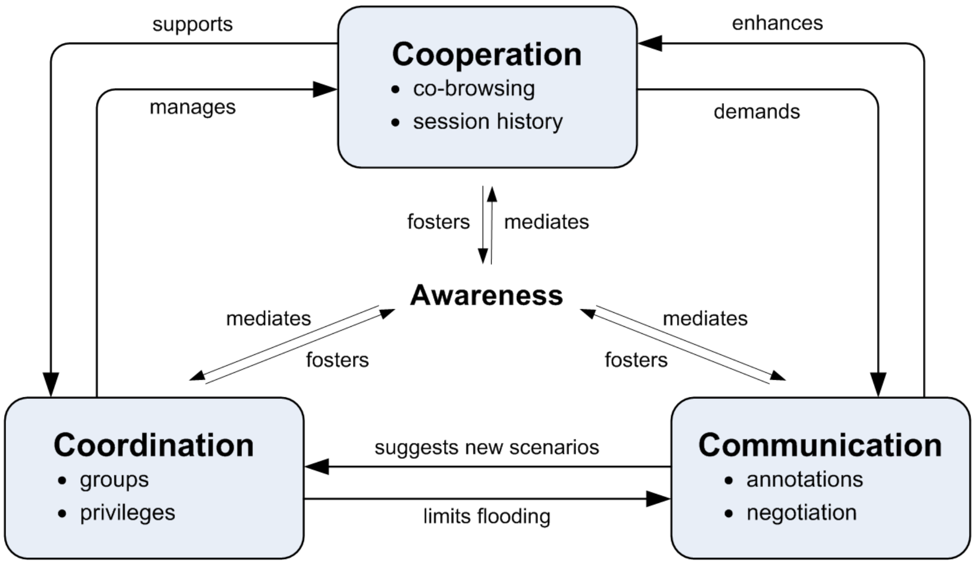


Figure 3.5: 3C aspects inter-relations inside OCEAN

The 3C-model, originally proposed by Ellis, Gibbs & Rein(17), has been usually applied

[4]We do not intend to provide complete analysis models in all terms of *software engineering*. Also, the formalization treated here focus exclusively on the co-browsing concepts, omitting system general concepts, for instance, users names.

on collaborative systems classification tasks (20), such as presented in Section 2.1.2. Even though, it is also important to use the 3C-model during the development process of groupwares (25). Considering this new approach, in the same way that (18), we explore this model as means to represent a co-browsing application domain and also to serve as a basis for groupware development.

Related work that defend such groupware development approach, however, do not offer any formal methodologies for supporting the entire development process of groupwares. Fuks et al.(18) propose a component-framework-based architecture for being used as skeleton for developing 3C-based collaborative services, thus helping on implementation of groupwares providing reusable components and architecture. But this proposal does not cover the whole process of developing a groupware. In fact, many other works as (21)(47) and (48), have proposed groupware development methodologies, based or not on the 3C-model, however the great majority have only focused on design and implementation issues, usually viewing groupwares as sets of plugged components.

In face of that, we propose a first step towards a comprehensive methodology for groupwares development. Such a step consists of providing support for initial stages of the software development process (prior to design and implementation phases), that are extremely valuable to the whole development (49). In this case, we are specifically focused on conceptualization tasks, being supported by a well-defined collaboration domain knowledge. Such conceptual modeling task followed an ontology-based approach, similar to the ones used on (50) and (49).

3.2.1 The Collaboration Ontology

The discipline of Formal Ontology has been employed in Computer Science noteworthy in three fields, namely, Knowledge Representation (within Artificial Intelligence), Database Systems and Software Engineering (51). In the later, ontology development has been taken as a means for *domain modeling*. This is meant to promote reusable conceptual models capable of facing the increase of size and complexity of software.

In this thesis we apply an ontology of the collaboration domain in this sense. The Collaboration Ontology (2) has been a source for the analysis development phase of the OCEAN system. This ontology has been proposed preliminary in (2), and is elaborated further in (19). We build the OCEAN conceptual models upon it, by extending concepts and relations that are present in particular within the collaborative web browsing application domain (52)(53).

The main benefits of such ontology-based domain modeling[5] are organized in as follows:

1. A domain ontology is supposed to be a strongly-axiomatized domain specification. As such, it is capable of restricting what can be said in specific applications within the ontology's universe of discourse. Consequently, if one takes such an ontology as a reference model for the analysis development phase of a given application (54), the conceptualization underlying the latter can hardly be subject of mistaken modeling decisions - the so-called *ontological adequacy* of the information system (54).

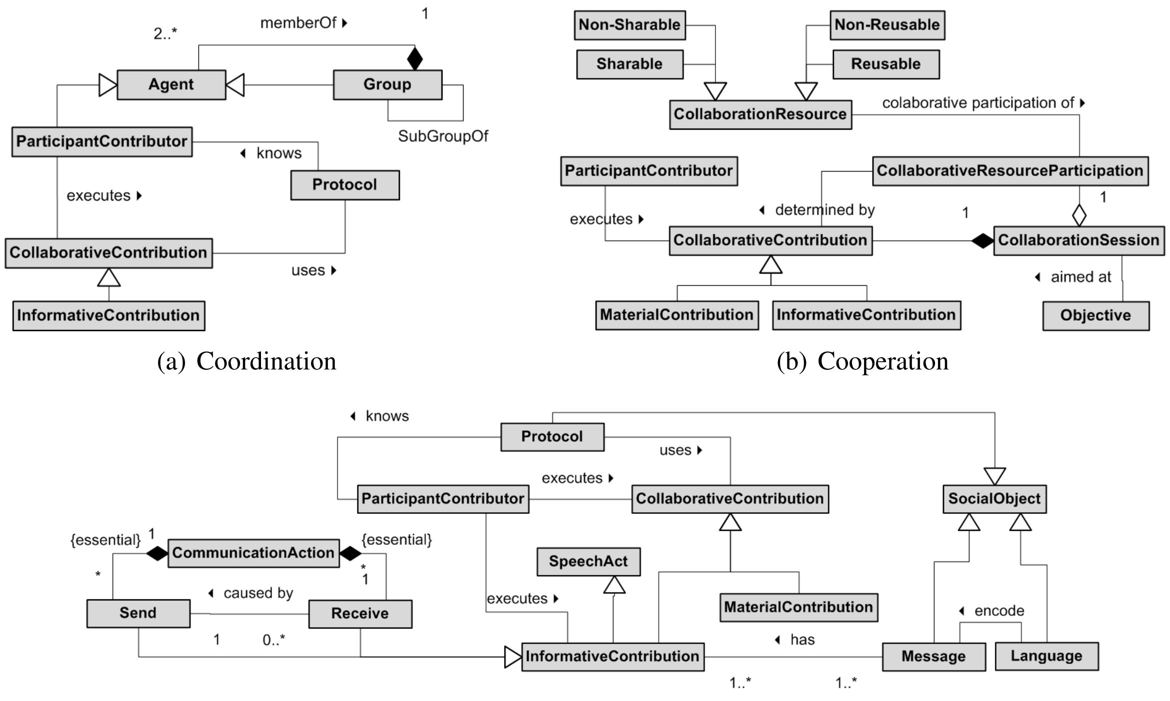
2. Domain (ontology) modeling can be an effective means for enabling the rapid prototyping of applications (54). Under this principle the latter no longer require an application-specific conceptual modeling effort from scratch. We have benefited from that in the development process of OCEAN as we take the Collaboration Ontology as a reference model.

3. As implicit in items 1 and 2 above, a domain ontology is supposed to be developed not to cover the scope of a single application, but rather to represent a subject domain in its essence (50). This broader perspective happens to be less biased, if at all, from technological issues which often restrict application-specific conceptual models. The reason is that it is grounded not in the symbolic world of information systems, but actually in the anchoring real world as we experience it. Such a deep-modeling effort can thus provide insight and input specially for (henceforth ontology-based) applications meant to interoperate with two or more other applications. This way we could, for instance, use such inputs for integrating OCEAN with general purpose communication tools, as discussed in Section 3.1.3.

Another point in favor of choosing the Collaboration Ontology, is that it has already been defined in accordance with the 3C-model. Thus, in an effort to a future 3C-model-based development methodology for groupwares, we propose the use of the Collaboration Ontology as a groupware conceptual modeling framework. As a consequence, groupware designers can have a well-defined start point for such development.

Fragments of the Collaboration Ontology, containing concepts related to the OCEAN context, are shown on Figure 3.6. Since this ontology is based on the 3C-model, it is also distinguished on coordination, cooperation and communication sub-ontologies, and its fragments are respectively presented on Figures 3.6(a), 3.6(b) and 3.6(c). The meaning of the relevant concepts for OCEAN, in accordance with (19), are described on Table 3.1.

[5]It is not the purpose of this thesis to elaborate on those assumptions since it goes beyond our scope. For a further reading, we refer the reader to (54).



(a) Coordination (b) Cooperation (c) Communication

Figure 3.6: Collaboration Ontology Fragments: fragments of each collaboration domain sub-ontology proposed by Oliveira(19)

.

3.2.2 OCEAN Conceptual Models

The Collaboration Ontology provides a conceptualization in an effort to cover the whole collaboration domain. Due to that, concepts, relations and attributes of the OCEAN conceptual models are mapped from concepts of this ontology. These concepts in most cases are specialized in order to achieve the specific application sub-domain needs. Such mappings are depicted by the concepts in *gray* in Figures 3.7, 3.8 and 3.9. The remain concepts (in *white*) remain to specialized characteristics of OCEAN's conceptualization.

First of all, the *co-browsing session* concept is the direct specialization from the ontology's *collaboration session* and represents all sessions in OCEAN. *co-browsing session* is the heart of OCEAN, holding all users interactions.

Regarding coordination aspect, the conceptual model depicted by Figure 3.7 organizes concepts related to groups establishment control and internal groups presenter-attendees protocol. The specific concepts introduced by this model (in *white*) are described in the following:

CobrowsingThread: as described in the previous session, every OCEAN's *co-browsing*

Table 3.1: Part of Terms Dictionary of Collaboration Ontology: the concepts present in the Collaboration Ontology (19) that are the most relevant to the OCEAN's conceptualization.

Concept	***Definition***
CollaborationSession	denotes an event in which participants interact for the purpose of collaboration.
ParticipantContributor	an agent that can contribute in a meaningful way to achieve the *objectives* of the *collaboration session*.
CollaborativeContribution	denotes an atomic event that one *participant contributor* executes in a *collaboration session*.
InformativeContribution	carry the information that is exchanged during a *collaboration session*.
CommunicationAction	denotes an act of communication between two or more agents.
Message	denotes the content of a participation of an agent.
Protocol	designates a set of rules which establish coordination for the harmony of the *collaboration session*.
Objective	denotes the motivation of the *collaboration session*, in others words, a reason that motivates its occurrence.
CollaborativeResourceParticipation	represents the participation of an object that has no expressed intention to participate, he participates in inanimate way (being used).
CollaborationResource	designates the object that can be either generated or consumed by the *collaboration session*.
Group	denotes a collection of agents, with a single identity criterion.

sessions are divided into sub-sessions, in an effort to provide coordination flexibility (see Section 3.1.1). Such sub-sessions are represented here by the *co-browsing thread* concept;

SessionGroup: remaining the Time/Space taxonomy(17), our proposal is a synchronous collaborative application. Due to that, knowing who are the online users is an important awareness information (26);

In this matter, *session group* represents the group formed by all *participant contributor* online in a same *co-browsing session*. Its main objective is to maintain session participants connected, even when they are divided into different sub-sessions (*co-*

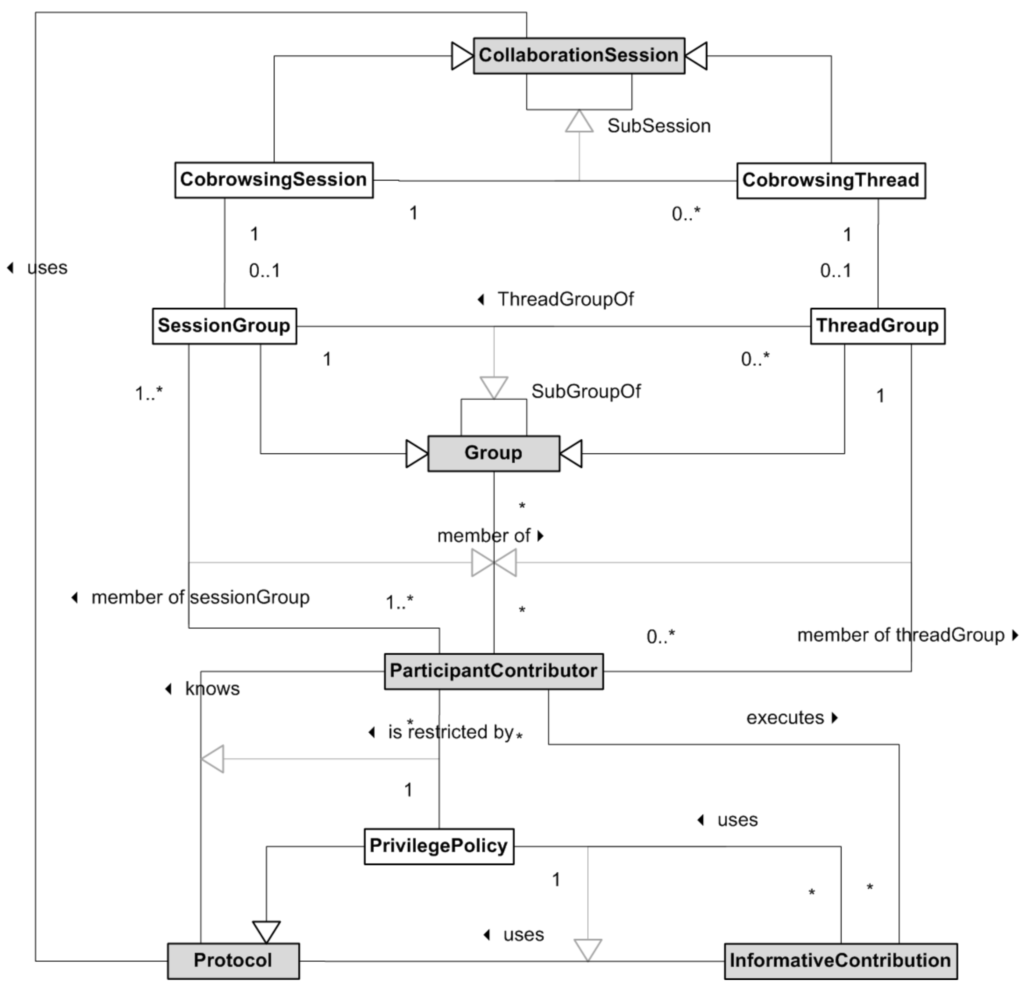


Figure 3.7: OCEAN Ontology-based Coordination Model: this model comprises concepts regarding coordination flexibility (*sub-sessions*) and simple management (*participants' privileges and coordination primitives*).

browsing threads);

ThreadGroup: a *thread group* also tracks online users, however considering specific *co-browsing thread* instances;

PrivilegePolicy: it is a specialization of *protocol*, and is responsible for all coordination actions inside OCEAN, managing the coordination primitives and participants privileges in the *co-browsing session*. More details about this concept can be found in section 4.2.2.

From now on we dissociate the co-browsing sub-session (*CobrowsingThread*) and the coordination subgroup (*ThreadGroup*) concepts, that were present as the same, during the previous section informal specification.

Regarding now the cooperation aspect, Figure 3.8 introduces OCEAN concepts which formalize the previously specified co-browsing history. In the sequence, such concepts are

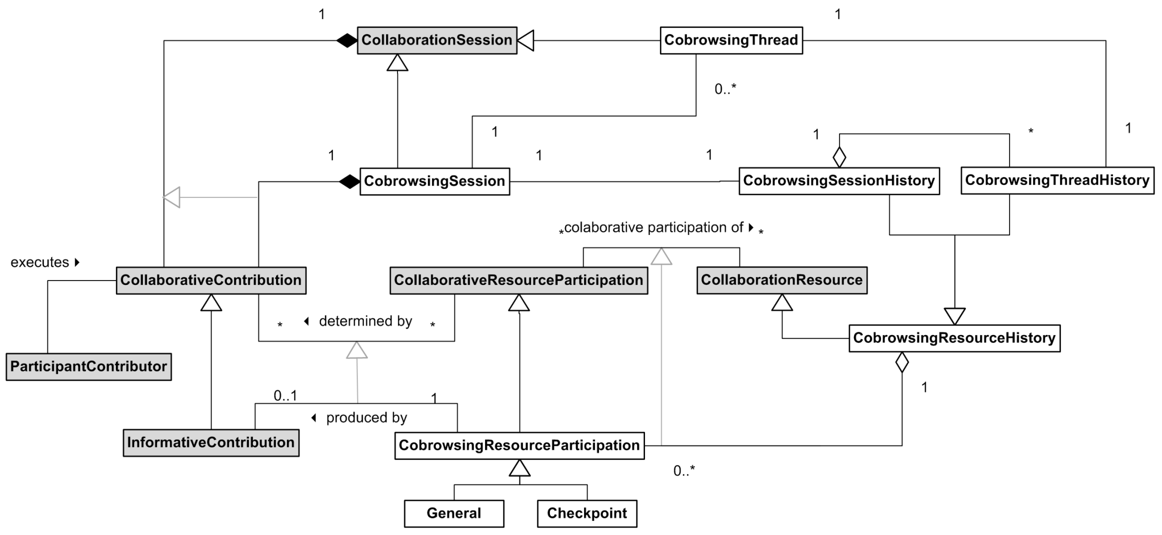


Figure 3.8: OCEAN Ontology-based Cooperation Model: this model comprises concepts regarding the management of the co-browsing history, either for its construction or revisiting activities.

described.

CobrowsingResourceHistory: a general registry of *co-browsing resource participation* instances, not providing any specific organization structure.

CobrowsingSessionHistory: it represents the OCEAN cooperation proposal, *i.e.*, a shared registry of the whole production of a *co-browsing session*. In other words, it is a specialization of *co-browsing resource history*, which stores a history of all contributions made in a *co-browsing session*, organizing this information in a multi-threaded timeline registry.

CobrowsingThreadHistory: following the cooperation proposal, instances of this concept comprises a linear registry of all *co-browsing resource participation* elements, that were produced by contributions inside the scope of one specific *co-browsing thread*. As a consequence, the *co-browsing session history* aggregates the particular sub-sessions' histories (*co-browsing thread history*) in order to have a complete registry of what happened everywhere in the session, at any time.

CobrowsingResourceParticipation: it is a specialization of *collaborative resource participation* which represents each atomic part of a *co-browsing resource history*. It can be compared to a "log entry"[6], recording an *informative contribution*.

[6]A single record involving details from one or more events and incidents. A log entry is sometimes referred to as an event log, event record, alert, alarm, log message, log record, or audit record. <*Common Event Expression*: cee.mitre.org/terminology.html>

Additionally, this concept can be classified as *general* or *checkpoint*. In short, a *checkpoint* is a kind of *co-browsing resource participation* that marks a change of co-browsing context, for instance, a browsing to a different web site. While *general* ones store all other type of contributions, for instance, a painted annotation.

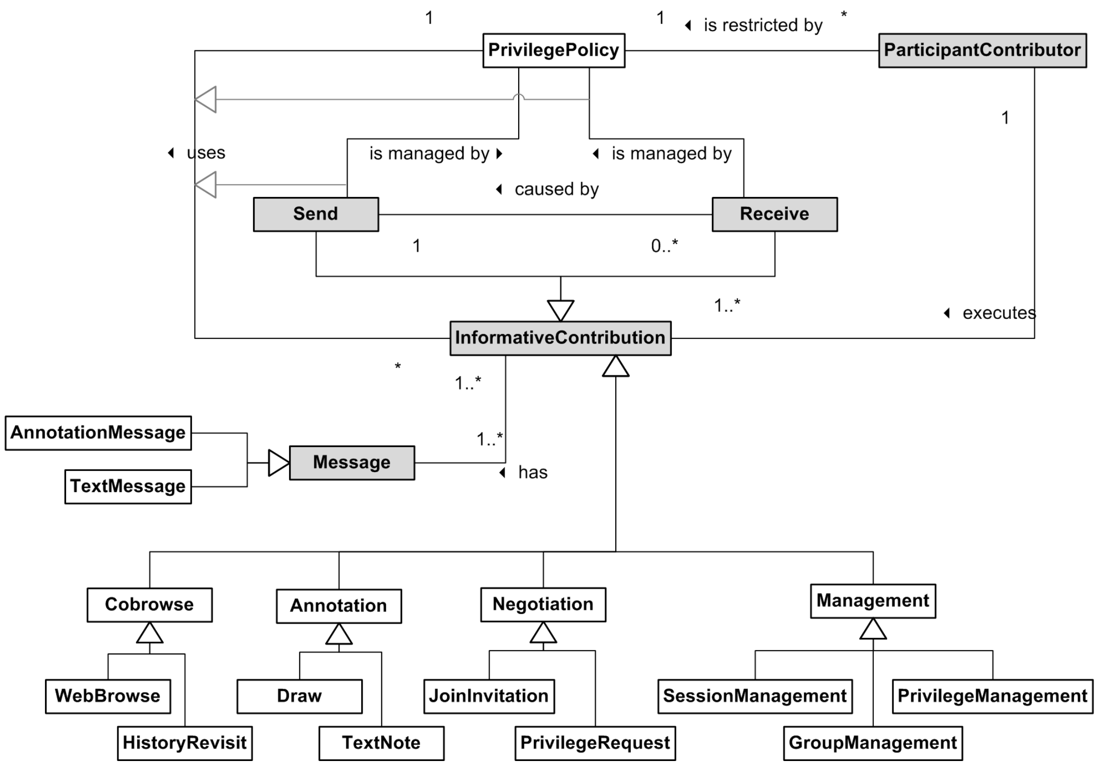


Figure 3.9: OCEAN Ontology-based Communication Model: this model comprises concepts regarding all communication actions between two or more participants of a co-browsing session. It includes pure communication as *annotations* and *negotiations*, and also the communication demanded for achieving coordination or cooperation tasks, e.g.: *privileges exchanges* and *history revisiting*.

Finally, Figure 3.9 presents the last part of OCEAN conceptualization, introducing communication related concepts, which are specializations of *informative contribution* ontology concept. They are used not only to classify what is being transmitted, but also to determine the nature of the transmission. For instance, who is transmitting is a relevant information available for these concepts, since there is the *executes* relation linking the *participant contributor* to the *informative contribution*.

Cobrowse: it is the specific type of *informative contribution* that handles collaborative browsing actions. In other words, *co-browse contributions* transmit what content the presenter wants to show to attendees. In particular, there are two types of *co-browse* contributions in OCEANconceptualization. Where *web browse* is the action of browsing to an Internet web page, while *history revisit* occurs when the presenter retrieves a *co-browsing resource participation* for showing some past co-browsing context to attendees.

Annotation: this concept represents annotation contributions. As defined, the annotations supported by OCEAN are geometrical strokes and small text notes, which are respectively formalized as *draw* and *text note* concepts.

Negotiation: it represents the negotiation facilities, which provide means of users suggesting new coordination scenarios. This concept's specializations stands for the two different negotiation primitives supported by OCEAN, *join invitation* and *privilege request*.

Management: the last *informative contribution* specialization represents the communication actions demanded to transmit session coordination decisions and state awareness. They are: (*i*) *session management*, which concerns any session state changes (*e.g.*, the login/logout of session participants); (*ii*) *group management*, which transmits group formation primitives; and finally, (iii) *privilege management*, which transmits privileges grants and revokes for participants.

These classifications do not describe what data is specifically being transmitted. We have specialized the ontology's *Message* concept onto two distinct message types. The *annotation message* is a structured definition of a painted annotation, specified in order to provide an easy transmission, storage and reproducibility. Whereas *text message* is a general purpose textual message type used for transmitting arbitrary information of the other contribution types, for instance, the URL transmitted by a *web browse* contribution.

3.3 Conclusions

This chapter has presented a comprehensive coverage of the proposal of this work, detailing what OCEAN intends to provide. In summary, OCEAN proposes a novel approach for collaboratively browsing the web, comprising characteristics of: (*i*) flexibility and manageability, with the advent of *co-browsing threads* and their coordination primitives; (*ii*) expressivity and dynamics, through the specialized communication features; and (*iii*) promoting a greater engagement on the teamwork through the use and reuse of artifacts collaboratively produced during the *co-browsing sessions*.

Such proposal characterization was entirely guided by the 3C-model (17), a relatively simple model originally proposed for classifying groupware (20). Moreover, following works like (18), we have adopted this model during this first phase of the OCEAN development process. This decision has contributed for this task, since having the distinction between cooperation, coordination and communication, we could exploit each aspect independently, thus focusing on proposing co-browsing solutions for each of them. After that, connecting each of these independent views rendered a complete and concise proposal for OCEAN.

Once our intentions were presented with OCEAN, this proposal has been formalized allowing to proceed to the development process with a consistent conceptualization of such intentions. In the same way, this formalization task followed the 3C-model, maintaining adequacy with the previous informal specification. However, the 3C-model itself is not enough for support this formalization, due to its extreme generality and lack of expressiveness. This way, we have adopted the Collaboration Ontology (2)(19), as pre-developed knowledge of the collaboration domain, built upon the 3C-model. Especially, this ontology-based conceptualization appears as an important contribution for the development of any groupware, since mirroring the Collaboration Ontology these systems could benefit of, for instance: avoiding mistakes in modeling decisions, rapid prototyping and easier interoperability with another collaborative systems. In face of that, we propose a first step towards a methodology for supporting the entire development process of groupwares, where the first phase is using the Collaboration Ontology for conceptualization.

Details about how each feature composing this proposal is designed and implemented are presented on next chapters.

4 Design Issues

The previous chapter discussed the proposal of this thesis itself, presenting a characterization and conceptual formalization of main features composing OCEAN. In this chapter, we are interested on *how* these features work internally. In doing so, the main concern now is related to what must be done to make this co-browsing system "idea" feasible in a real scenario, in this case, the Internet.

Such discussion mainly includes, understanding the challenges involved for providing those features, especially co-browsing facilities, as well as on the required infra-structure for supporting these features. As a result, we aim to present here the design[1] of OCEAN, which is a step further towards the realization of this work proposal.

Two requirements, in particular, have concentrated our attention during the design task, namely flexibility and performance. Firstly, we aim at proposing a design that could be flexible enough for easily supporting reorganizations on the collaborative work. In addition, such flexibility comprises an extensibility aspect, regarding a support of this design for some further improvements on the proposal. At last, we also have a great concern on providing a good performance for our co-browsing service, considering mainly responsiveness and scalability.

This way, the remaining of this chapter is structured according to the following organization. Section 4.1 positions our proposal among common distributed architectures adopted by other co-browsing systems, presenting the involved entities distributed over the Internet and its main components. Section 4.2 shows an application level network protocol, proposed for providing communication among such distributed entities. At last, Section 4.3 gathers some conclusions about OCEAN's design.

[1]We do not intend to provide a complete software design description and modeling in terms of *software engineering*. We focus on describing what we consider relevant contributions to the collaborative web browsing field.

4.1 Distributed Architectures

The major advantage of using a co-browsing system is that it allows distant users to jointly navigate the web, instead of just supporting web sites recommendations and discussions through standard communication tools. Such co-browsing system's conception must rely on a distributed architecture that offers as less overhead as possible in order to not disturb users' collaboration. Thus, we have designed a distributed architecture based on a lightweight infra-structure for supporting co-browsing sessions. Here we present such architecture, which comprises independent entities distributed over the Internet. Also, each entity is detailed, showing its internal functional components.

Before presenting the OCEAN is architectural proposal, we discuss about related architectures commonly adopted by other co-browsing services, and the problems faced by them, due to their architectural choices.

4.1.1 Commonly Adopted Approaches

Other co-browsing initiatives have proposed particular solutions for providing their features to distributed users. Thus, there are similarities among them, probably due to the fact of having faced the same specific challenges on such development. In a first glance, the most basic entities involved in any distributed co-browsing session certainly are:

- *Users*: the people that have met intending to co-browse. In other words, the participants of the co-browsing session.
- *Co-browsing User Application*: considering that users are located at different places, they need an computational interface in order to interact each other through the co-browsing session. Such interface usually is referred as *co-browsing clients*, implemented software running in computational device (client host) in pose of users. In fact, this client represents the proper user in the co-browsing sessions.
- *Shared Contents*: this entity represents everything that is shared in a co-browsing session, where the most relevant are the co-browsed web pages.
- *Web Servers*: in most of cases, those shared web pages are published somewhere in the Internet. This web pages' sources are the *web servers*, hosts dedicated to serve pages, thus composing the proper Web.

The client application is an entity that acts producing and consuming information from the co-browsing session. Usually, co-browsing systems implement this entity following one of two basic approaches: (*i*) implementing a specific collaborative browser application for clients (22)(5); and (*ii*) using standard browsers to access an implemented co-browsing service (1)(12).

When following the first approach, co-browser designers are free to implement whatever they want. For instance, they are able to choose any network communication protocol since they have direct access to the clients' operating system. The main drawbacks are, however, that building a fully fledged browser is a gigantic task, and moreover having to use a special browser in some context, and a standard browser in others, can be very inconvenient for the users (8).

On the opposite, when using a standard browser, the designers take advantage for instance of, security updates and already implemented content rendering mechanisms. Besides, the greatest advantage of using a standard browser is that users are already familiar with its interface, and so, feel comfortable on using this browser for a co-browsing activity. In particular, some initiatives follow a hybrid approach, extending a standard browser in order to cope with a co-browsing service. An example is the CoBrowse(29), which extends the Mozilla Firefox, adding collaborative browsing capabilities to this standard browser.

Considering arbitrary users collaboratively browsing with arbitrary goals, the standard browser usage approach is preferable for designing an adequate co-browsing system. Actually, it is the most adopted by other co-browsing systems, for instance, (30)(14)(10)(6) and (12). Usually, co-browsing systems like these are designed as *dynamic web pages*[2] which somehow act as co-browsing client applications, managing access to the co-browsed web pages and all shared content. However, all co-browsing services that were designed to use a standard browser this way, have faced at least a common problem, the native security issues. These standard browsers usually restrict any dynamic code (quite often represented by scripts) of accessing information from different web domains. In other words, a co-browsing management script running in the co-browsing client (dynamic page) cannot access information from the shared content (co-browsed web pages), when these two web pages (co-browsing client and shared content) are from distinct domains (usually different web servers).

Note that, when both web sites are from the same domain, the browser does not impose any restriction. Thus, there is no problem to add co-browsing capabilities to a web portal, or to provide a co-browsing service inside an Intranet, that is the case of the co-browsing

[2]Classical hypertext navigation occurs among *static* documents, and, for web users, this experience is reproduced using static web pages. However, web navigation can also provide an interactive experience that is termed *dynamic*. Content (text, images, form fields, etc.) on a web page can change, in response to different contexts or conditions. There are two ways to create this kind of interactivity: client-side scripting and server-side scripting. <en.wikipedia.org/wiki/Dynamic_web_page>

systems presented in Section 2.3.2. For instance, if a news portal (hosted on: `http://DOMAIN/`) would wish to offer an specialized co-browsing system, its maintainers just need to publish the co-browsing dynamic page at the same domain (`http://DOMAIN/cobrowse_news`). However, such solutions restrict too much the usages of general purpose co-browsing (see section 2.2), since in this scenario users cannot leave this single domain (in this case *DOMAIN*), for collaboratively checking more information at the huge source that is the Web.

Avoiding domain-specific co-browsing applications, standard web browsers impose restrictions on different domains, since this "cross access" is considered a security breach. In order to overcome such a problem, thus allowing users to freely co-browse thorough arbitrary domains, some co-browsing systems have proposed mechanisms for circumventing browsers security restrictions. The most commonly adopted of these mechanisms are subsequently described.

Content Manipulation: Most co-browsing systems in the literature are based on a specialized proxy-server, that intermediates the transmission of web pages for the referred web servers to all session members, e.g.: (55)(31)(14) and (56).

This proxy translates every accessed web page, modifying its original content, in an effort to overcome browser security. This can be made basically by two ways. One is changing the domain of the shared content to the same domain of the co-browsing service (a kind of *domain spoofing* or *faking*). Therefore, the browser would allow the co-browsing client scripts to access shared web page information(10). The second way is embedding event listeners in the shared web page, so these events aware web page can detect its own events and cope with the co-browsing system (30). However, these translation techniques can negatively affect the proper shared web page, for instance interrupting original scripts' behavior. Such scripts might not expect that the containing document has been "moved" to a new domain, or that a new listener are modifying a normal web site behavior.(12).

Also, the proxy architecture leads to potential performance problem, since the proxy server can become a bottleneck (11). Indeed, proxy-based co-browsing architectures cause overhead on the server side, due to the translation and redistribution of the whole shared web pages for all clients.

Browser Manipulation: This mechanism consists to overpass the browser security from the inside. It can be made running security authorized embedded applications in the standard browser during co-browsing sessions. Common alternatives use Java applets or specific browser extensions (29). Once inside the browser, these codes are free to grab whatever information they want, from any browsed web page.

> Because of the insecurity or unreliability of such applications, some browser's users do not like to install browser extensions or to run authorized Java applets. However, there are certification mechanisms, for instance, provided by the Mozilla Co. that aggregates a reliability degree to an browser extension. Even so, some statistics indicate that users usually don't care about it. Statistics regarding the Mozilla Firefox browser can be found at the official blog of Mozilla Add-ons[3], blog of metrics[4] and at Mozilla Add-ons statistics dashboard[5]. As an example, at November 19th in 2008 the Mozilla.org have served their billionth add-on download since 2005, and only in one day (April 27th, 2009) this same web site have registered 1,919,221 downloads, where a great amount of them are not signed.
>
> Finally, this approach avoids synchronization problems in the co-browsing sessions, since users are independent to download shared web pages directly from their source (web servers). Therefore, users can leverage their own bandwidth, no longer depending on the connection to a specific intermediary proxy server. Another benefit of this approach is that it does not affect web sites' internal scripts, unlike proxy-based co-browsers, which usually change web sites' URLs and internal script listeners.

Another point worth to consider is how the co-browsing clients communicate. It is necessary that these applications frequently exchange messages in order to keep users aware of everything that happens in the co-browsing session. Considering that these clients are dynamic web pages running in a standard browser, the naturally available communication mechanism is only HTTP (Hypertext Transfer Protocol) connections[6] .

This way, it becomes necessary an HTTP server (web server) for intermediating such communications. Especially, proxy-based systems reuse this same proxy for also intermediating client communications. Besides convenience, such approach overloads even more the server, that has already to deal with web pages translations and distributions. In particular, another network protocols can be used when the co-browsing client copes with a Java applet or a embedded Flash. In this case, these mini-applications are able to manipulate sockets for instance, thus being capable of using another application level protocols rather than HTTP. In doing so, such co-browsing clients could communicate though more elaborated mechanisms, for instance in a peer-to-peer network or using multicast channels.

[3]Official Blog of Mozilla Add-ons: blog.mozilla.com/addons

[4]Blog of Metrics: blog.mozilla.com/metrics

[5]Mozilla Add-ons Statistics Dashboard: addons.mozilla.org/en-US/statistics

[6]Web browsers are essentially HTTP clients, applications designed for retrieving and rendering web pages from HTTP servers. This way, the communication mechanism that these browsers usually offer for such dynamic web pages' scripts are generally all HTTP based.

4.1.2 Our Approach

Due to the presented advantages, we have also chosen to use a standard browser as the OCEAN client. Similar to other co-browsing approaches(14)(10), this standard browser accesses our *co-browsing service*, which provides a dynamic web page that is our *client application*. This client manages every contribution made by its user, as well as maintaining session awareness.

An architecture that provides better performance, concerning for instance, scalability and responsiveness, we have designed OCEAN following a *browser manipulation* approach. Thus, one of OCEAN's requirements is that clients use a browser extension instead of being connected to a web proxy.

The additional entity composing our architecture is responsible for keeping the client applications in touch, broadcasting contributions to session participants. In order to make it simple, it is designed as a centralized service built upon an HTTP server, which acts as a mediator for client communications. Due to its centralized nature, the OCEAN application server could also become a bottleneck for the whole system. However, despite proxy-server solutions that have to handle shared web pages' content, our server must handle only small pieces of data, representing participants' *informative contributions* (see Section 3.2), where great part of the data is treated only in the client application. It means that our server works basically like a message router, receiving and re-transmitting small data packets. It can be seen by the proper *web browse informative contribution* (a co-browsing action). In OCEAN, a co-browsing action is supported by sharing *URL*, which identifies the co-browsed web page, being the clients responsible for retrieving the related content from its original web server. Whereas, besides handling URLs, proxy-based architectures also have to handle the contents of all co-browsed web pages in server-side.

So, in summary, comparing the performance of OCEAN application server with a common proxy-server, OCEAN has two main advantages. The first concerns server resources saving, as it must process a much smaller amount of data not having to translate any web content. Secondly, network connections consume less bandwidth, since OCEAN server only handles transmissions of small session events.

In the same way that a *browse contribution* which transmits a URL message for archiving a co-browsing action, the other types of *informative contributions* defined at OCEAN conceptualization also are based on small message transmissions Such small messages do not demand too much effort to be transmitted, which does not overcharges the server connectivity. On the contrary, proxy-servers usually have to download the web page contents from its web server and then redistribute it to all interested session members. This mechanism could quickly overcharge the proxy-server connectivity when the number of interested members (session participants) or simultaneous sessions grows up, consequently compromising its scalability, and thus the whole co-browser application performance.

Another point concerns server geographical position. Such positioning may considerably affect shared web content download time. As an example, imagine a scenario where a co-browsing server is located in an arbitrary European country, and all participants of a session are located in Brazil, and they are co-browsing Brazilian web pages. If the application server acts as a common proxy-server, for every browsing, the European proxy downloads the web page content from Brazilian web servers and , after that, the Brazilian users download this content from the European proxy. Note that, proxy-server solutions thus impose propagation delays to every web page download. Conversely, on OCEAN, the client application performs web page downloads by their own, not depending on the application server for this task. In OCEAN, such propagation delays also affect the distribution of *informative contributions*, however we demonstrate in Chapter 6 that these delays are irrelevant in the context of the whole session.

As a result, OCEAN was designed to be a lightweight system, dealing with three distributed entities, the *application server*, the *client standard browser* and the *web servers* which hold the co-browsed web pages. The application server acts as a synchronization point, dealing with a minimal workload to accomplish its task. On the other side, the client has greater responsibilities, in order to save server resources. Figure 4.1 depicts an architectural overview of OCEAN, presenting the distributed entities and their main internal components.

The application server is essentially composed by the *notification service*. This service is responsible for handling all the created contributions, forwarding them to the concerned participants. Such contribution routing process is coordinated by the *notification protocol*, explained in the next section. In addition, the application server is composed by the *session manager*, that controls all co-browsing sessions in the system, managing for instance, the participants privileges. At last, the *co-browsing history repository* is used for storing all the information about all co-browsing sessions, composing the co-browsing histories. As described on Chapter 3, such histories are a persistent registry of the session contributions, important for maintaining a common workspace for all participants, even when working on separated co-browsing threads.

At the client side, the participant uses a standard browser accessing the OCEAN *client application*, that in turn, encapsulates the shared contents (web pages and other contributions).

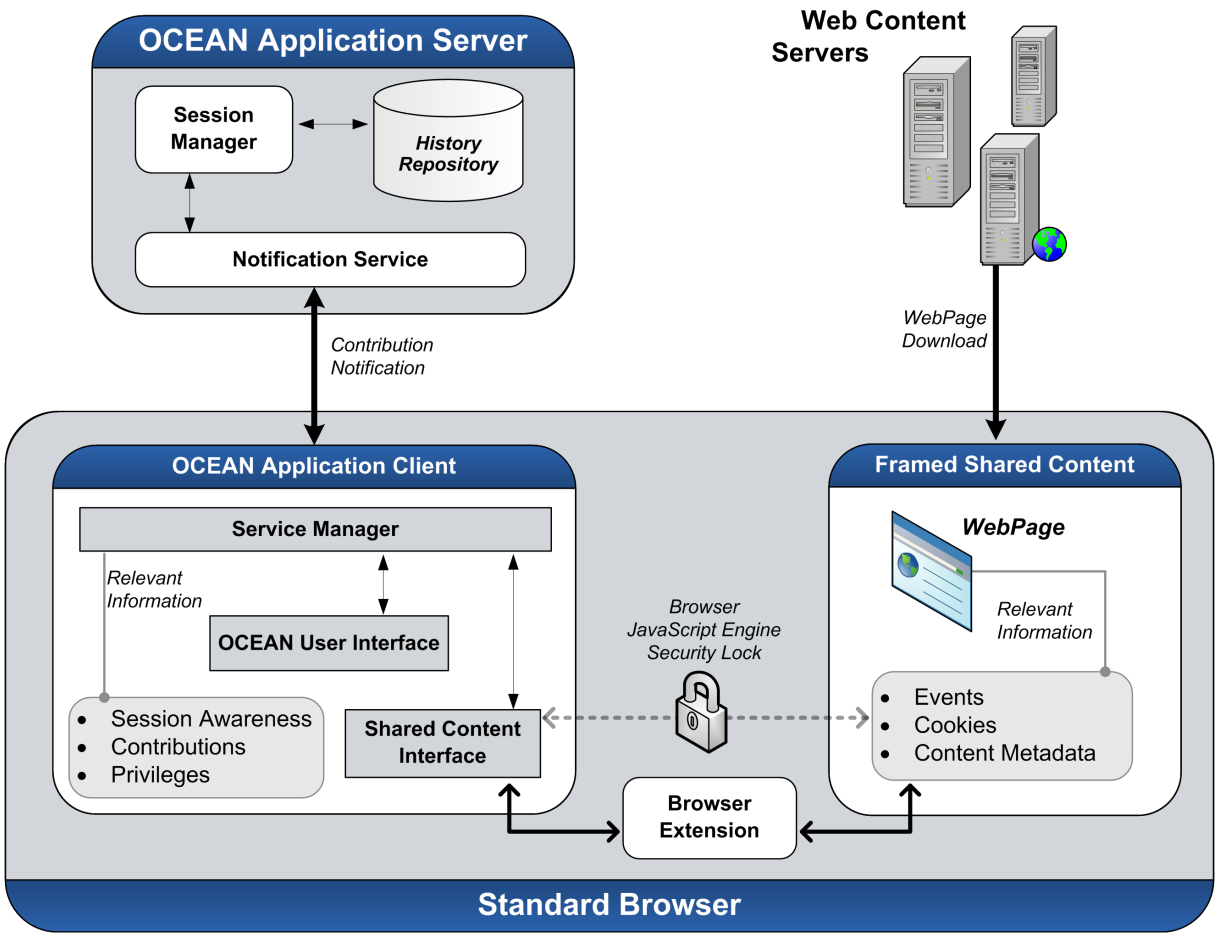


Figure 4.1: Distributed Architecture Components: presents the distributed entities involved on the OCEAN co-browsing activity, detailing its main components.

Due to standard browsers security constraints, this architecture rely on the *browser extension* component, working as a access bridge to shared content relevant information. Nevertheless, the *browser extension* is not a mandatory requirement for using OCEAN. For example, if a user does not want to install it, thus he simply could not act as a *presenter*. Even though, he/she still would be able to participate of a group as an *attendee*.

In particular, OCEAN *client application* is composed by three main components. The *service manager* is the core module. Its main responsibilities are: observing user actions, restricting the unprivileged ones; handling created and received contributions; manipulating session awareness state; and organizing user's interface events. The *shared content interface* is responsible for observing and restricting user actions inside the encapsulated shared content, in accordance with the *service manager*. OCEAN *user interface* is in charge of presenting all the related information to the user, and providing system interaction features, for instance creating annotations or defining co-browsing threads.

It is important to mention that, with the use of the *browser extension*, OCEAN could also share *HTTP cookies*. Cookies are parcels of text sent by a web server to a client (usually a browser) and then sent back unchanged by the client each time it accesses that server. They are usually used for authenticating, session tracking (state maintenance), and maintaining specific information about users, such as site preferences or the contents of their electronic shopping carts[7]. In a co-browsing session such cookie sharing feature is valuable for maintaining all clients in the same state with the shared content's web server, as if they were only one user. However, sharing cookies implies on a privacy discussion(57). In this matter, OCEAN actually only warns the group presenter every time the application client is sharing his or her cookies. In future works we intend to offer some interactivity or manageability for this cookie sharing mechanism.

4.2 Notification Protocol

We have defined in OCEAN's conceptualization that, each user interaction in a co-browsing session implies on *informative contribution* transmissions. Each transmission is represented by a user sending a message and, after that, at least one different user receiving it. In order to avoid lost of information and to keep sessions consistency, OCEAN provides a standardized communication protocol for supporting such transmissions. This protocol also embeds processes related with each *informative contribution* act, for instance, disallowing unprivileged users to send messages.

The messages handled by our system consist of small amount of data describing the supported *informative contributions*, which regards to browsing, annotation, negotiation and management actions. For example, a *web browse* contribution is described by a message containing only the URL of browsed content. Hence, in a nutshell, the communication protocol consists of small messages distribution over the participants of a session. As the notification protocol is responsible for notifying, or publishing, some eligible consumers about produced contributions, it has been specified as a specialization of the Message Queues (58) and Publish/Subscribe (58) communication paradigms .

According to Eugster et al.(58), message queues provide global spaces, which are fed with messages from producers, being concurrently pulled by consumers afterwards. The main characteristic is that message queuing systems usually provide transactional timing and messages' ordering guarantee. On the other hand, in publish/subscribe paradigm, messages (or events) are published by the producer, and after that, these messages are pushed to the

[7]Wikipedia - HTTP Cookies. <en.wikipedia.org/wiki/HTTP_cookie>

consumers who have previously subscribed to referred message's class. The different ways of specifying the events of interest have led to several subscription schemes, where the most widely used are: *topic-based*, *content-based* and *type-based*. Although, message queuing and publish/subscribe are tightly intertwined: message queuing systems usually integrate some of publish/subscribe-like interaction.

Our protocol inherits from message queues the notion of a shared and global ordered messages/events space for the session. This characteristic allows us to maintain the cooperation history scheme proposed on Section 3.1.2. Considering topic-based publish/subscribe paradigm, our protocol inherits the decoupling interactions and events interest matching on subscribe action. However, there is a great difference if compared with traditional publish/subscribe approaches. In our case, the subscribers are not free to explicit us determine which events interest them. These interests are automatically determined by OCEAN, in accordance with the session's coordination scenario (see Section 3.1.1). In addition to that, the rights to publish any event are also coordinated by this same scenario.

By definition, in the topic-based publish/subscribe scheme, participants can publish events and subscribe to individual topics, which are traditionally identified by *keywords*. Although, topics are strongly similar to the notion of groups, where groups represent disconnected event spaces. Consequently, subscribing to a topic *T* can be viewed as becoming a member of a group *T*, and publishing an event on topic *T* translates accordingly into broadcasting that event among *T* members (58).

Therefore, considering minor adaptations, this paradigm fits nicely our communication needs. For each collaboration session (*co-browsing session* or *co-browsing thread*) in OCEAN, a new publish/subscribe topic is established, where the associated *keyword* is the sessions' *unique ID*. As a consequence, whenever a participant for instance joins a *co-browsing thread*, he or she automatically becomes publisher and subscriber of that topic. Consequently when leaving this thread, the participant ceases to be publisher and subscriber of such topic. Also, when a presenter, for instance, publishes an *informative contribution*, all the attendees (subscribers of such topic) would automatically receive this contribution.

4.2.1 Definition

This section describes the main concepts composing the *Notification Protocol*, highlighting their main characteristics. Figure 4.2 depicts such concepts and the relations between them, extending OCEAN conceptual models. The concepts and relations introduced here are described in the following, while the others are presented in Section 3.2.2.

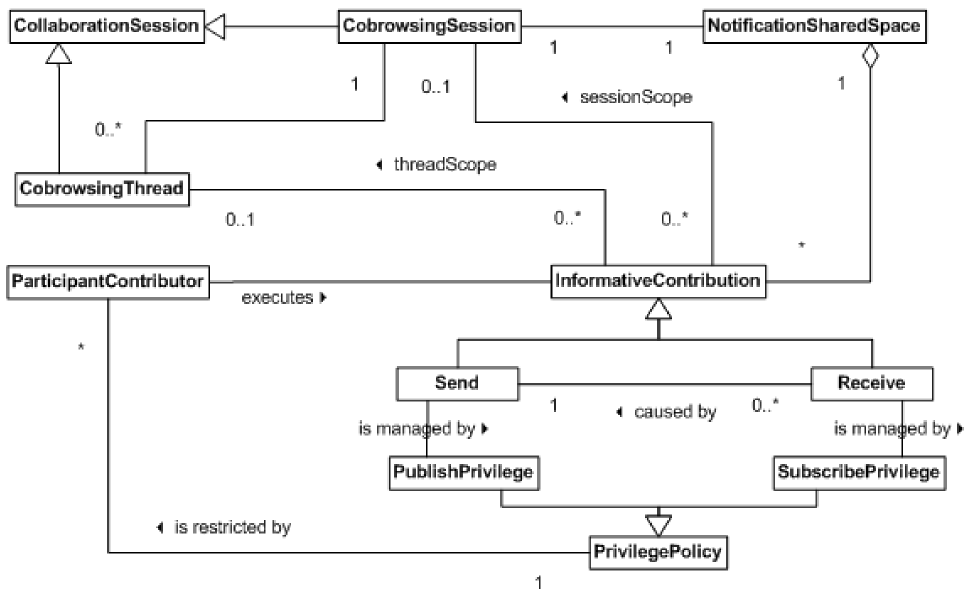


Figure 4.2: Notification Protocol Design Concepts: a model extending OCEAN conceptual models (Section 3.2). It introduces concepts related to how the informative contribution are handled by the system.

NotificationSharedSpace This concept represents the shared and global ordered space of messages/events for the session. This message queue like structure maintains all the published *informative contributions* instances, while there are participants meant to receive them. Meanwhile, these contributions produces their respective history artifacts, which are stored at the *co-browsing history repository*. After being correctly distributed and stored, the *informative contribution* instance is destroyed, discharging the *notification shared space*.

Moreover, this structure is designed as an unique queue of contributions per co-browsing session. Every queued contribution are labeled with the destiny scope. Thus, during the notification process delivering phase, the *notification service* can determine who would be the eligible subscribers for such information, based on the privileges policy.

Scope Relations These relations (*session scope* and *thread scope*) represent topics in terms of the publish/subscribe topic-based paradigm. In this matter, the participants can publish and subscribe for specific topics. So each collaboration session in OCEAN is a publish/subscribe topic in the *notification protocol*, either *co-browsing sessions* or *co-browsing threads*.

The *session scope* provides a communication channel for session's participants to exchange information, independent of specific thread group[8] they belong to. Such topic

[8]*Thread group* is the group of online users participating in the same co-browsing thread. (see Section 3.2.2)

is used to keep participants aware of any modification on the hole session state or configuration. For instance, informing users' login and logout, and also informing group reconfiguration. More important, providing groups configuration awareness is a valuable social navigation (9) feature. For example, while observing that many users are migrating to a specific group, a observer may conclude that something interesting is happening in such group, without the need of any formal invitation from other users.

The *thread scope*, in turn, permits message exchanging in the context of a specific co-browsing thread, which supports interest matching considering the threads formation. This way, messages exclusively from a thread do not annoy other threads' participants with unnecessary notifications, who might not be interested on the referred information that time. Even though, if these other participants would be interested, they still could access that information at the *co-browsing history*.

PublishPrivilege Every action that a user can perform in OCEAN is controlled by privileges. So, for every participant, privileges are granted and revoked in order to coordinate "*who can do what*" in the session. This way, *publish privilege* is a specialization of *privilege policy* concept, and represents the privileges for determining if a user can or cannot publish some information on a specific topic.

SubscribePrivilege In the same way, the *subscribe privilege* represents the privileges for determining if a user can or cannot receive some information from an specific topic. Note that, this concept is actually the definition of OCEAN's automatic subscribe action. In other words, if a user has granted, for instance, the *subscribe privilege* for receiving annotations from an arbitrary co-browsing thread *T*, he/she is automatically a subscriber of annotations events on topic *T*. Therefore, this concept is what most differ OCEAN notification protocol from traditional publish/subscribe communication paradigm.

These concepts present the nature of our notification protocol. They figure out the inherited characteristics from traditional protocols, and what we have introduced for our specific context, on *publish privilege* and *subscribe privilege*. It is worth mentioning that our protocol inherits another characteristic from modern topic-based publish/subscribe approaches, namely the *hierarchical addressing*. Such feature permits programmers to organize topics according to containing relationships (58). In this matter, OCEAN also organizes its publish/subscribe topic hierarchically, where on *session scope* comprises a set of *thread scopes*. Especially, such hierarchy mirrors the proper OCEAN conceptualization, considering that one *co-browsing session* can be composed by many *co-browsing threads* as sub-sessions.

4.2.2 Privileges Management Policy

In an effort to maintain the coordination inside OCEAN, privileges were designed for allowing or disallowing users to perform collaborative actions, in other words, to coordinate sending and receiving of *informative contributions*. These privileges associations are determined combining 3 (three) different parameters, subsequently described.

- *Scope* - the scope parameter defines how wide is the interaction in the session and is defined by the notification protocol scope relations, *i.e.*: *session scope* and *thread scope*. By definition, the most widely available scope for users is the *session scope*, since two different co-browsing sessions are considered independent and isolated. In other words, participants from one session are not allowed to directly contribute to other sessions' participants. Such collaboration scenario could only be possible through the retrieving of other session's history. Following the topics hierarchy, the *thread scope* provides a specific interaction scope;
- *Function* - this parameter remains to the *informative contribution* concept specializations denominated *send* and *receive*. Such concepts represent the two main phases of the notification process, linking who are the publisher and subscribers of such interaction;
- *Type* - at last, this parameter denotes the all possible message types handled by the notification protocol. These messages are classified as follows: (i) Session Management, including *login* and *logout* messages; (ii) Coordination Messages, which includes *create thread*, *join thread*, *leave thread*, *token passing*; (iii) Communication Messages, with *web browse*, *annotation*, *privilege request* and *join invitation*; and finally (iv) Cooperation Messages, that contains only *history revisit*.

In order to illustrate the OCEAN privileges management policy, we have fixed the scope and varied the other two parameters. In doing so, Table 4.1 shows the most relevant privilege grants for the *session scope*. All other contribution types not present in this table are considered *revoked* on both functions. In particular, these privileges configuration for the session scope are invariable during all session long. Moreover, the *login* contribution is the only one revoked, because this contribution is automatically published by the system, and not by the user.

In the same way, Table 4.2 presents the *thread scope* privileges policy. Despite static configuration of *session scope* privileges, *thread scope* privileges can be dynamically changed. The first change cause are coordination primitives (*i.e.*: *create thread*, *join thread* and *leave thread*). The *create thread* primitive starts a co-browsing thread, demanding the initialization

Table 4.1: Session Scope Privileges

Contribution Types	**Publish Privilege**	**Subscribe Privilege**
Login	`revoked`	`granted`
Logout	`granted`	`granted`
Create Thread	`granted`	`granted`
Join Thread	`granted`	`granted`
Leave Thread	`granted`	`granted`
Join Invitation	`granted`	`granted`

of privilege policy structures. On the other hand, *join thread* and *leave thread* primitives cause respectively the inclusion and exclusion of a participant from a thread group. Inclusion actions leads to the re-configuration of included participant's privileges, whereas an exclusions leads to two possible consequences. At first, if the user who leaves was an attendee, this action only affects this specific user, revoking his/her privileges for the related *thread scope*. However, if this user was the current co-browsing thread presenter, besides revoking his/her privileges for that *thread scope*, another user inside this thread must be elected the new presenter. In this case, OCEAN just automatically grant presenter privileges to the first available participant in the thread group, after that, participants can re-negotiate in order to choose a new presenter by their own.

Table 4.2: Thread Scope Privileges

	Presenter Role Privileges		***Attendee Role Privileges***	
Contribution Types	*Publish*	*Subscribe*	*Publish*	*Subscribe*
- Browse	`granted`	`revoked`	`revoked`	`granted`
- Annotation	`granted`	`revoked`	`revoked`	`granted`
- Cookie	`granted`	`revoked`	`revoked`	`granted`
- Presenter Privilege Request	`revoked`	`granted`	`granted`	`granted`*
- Download Ack **	`revoked`	`granted`	`granted`	`revoked`
- Presenter Token Pass	`granted`	`revoked`	`revoked`	`granted`
- History Revisit	`granted`	`revoked`	`revoked`	`granted`

* Useful for maintaining the whole thread group aware of someone's intention to be a presenter.

** Introduced in Section 4.2.4.

Another cause of change in a *thread scope* privilege grants is when the presenter ceases the presentation privileges to an attendee, exchanging their roles in the session. Such agreement is supported by the coordination basic primitives: *privilege grant* and *privilege revoke*. Note that operation of exchanging the presenter-attendees roles leads to a set of these basic primitives calls. Considering that presenter-attendees role exchange is a very common operation, OCEAN defines a composition of the primitives calls in order to provide role exchange as a unique action, denominated *presenter token pass*. Such shortcut operation creates the idea of "material" and unique token per group, where only the user who hold this token, has full rights to present. Such token based coordination mechanisms are usually referred as floor control policies (59). Following a pseudo-code format, the Algorithm 4.1 describes presenter token pass operation, showing the calls of privilege grants and revokes, demanded to exchange two users roles.

As a consequence of such tokens, it becomes necessary to specialize the *privilege request* contribution into *presenter privilege request* and *annotation privilege request*, where the first has already been presented in Table 4.2. The second follows the same logic. In this case the annotation token holder has his/her privileges set as revoked and granted for the publish and subscribe functions respectively, while the others have granted and revoked for the same respective functions.

A remark is that OCEAN currently does not support two participants of a same group creating annotations at same time. This restriction was imposed in order to simplify the consistency management of this feature. However, considering flexibility objectives of our proposal, we have extended the privileges policy, creating the *annotation token*. In the same way that with *presenter token*, the *annotation token* is a shortcut for exchanging privileges. In this case, only privileges regarding the annotation contribution type are handled. Accordingly, one user can keep presenting the web content while ceases the annotations ability to an attendee. The annotation token pass operation is described in Algorithm 4.2.

4.2.3 The Notification Process

Up till now we have described a general view of OCEAN's Notification Protocol, focusing on related concepts and rules. Now we focus on the notification process, considering what happens to completely transmit an *informative contribution* from its producer (publisher) to all the concerned consumers (subscribers).

According to the protocol definition, every *informative contribution* inside OCEAN's environment is transmitted through the Notification Protocol. This protocol relies on a central *notification service* for intermediating such interactions, maintaining producers and consumers loosely coupled. With this in mind, the implied notification process is composed mainly by two independent phases. The first phase regards the producer publishing a contribution to the mediator (*notification service*), while the second phase regards the mediator delivering this contribution to the privileged consumers. Also, these transmission phases incorporate session coordination and system management activities, in order to maintaining the consistency of the ongoing co-browsing session.

Algorithm 4.1 Presenter Token Pass Operation

```
Require: $user1 \neq user2$
Require: $user1 \in G1 \wedge user2 \in G1$
Require: $isPresenter(user1) \wedge isAttendee(user2)$
Ensure: $isPresenter(user2) \wedge isAttendee(user1)$

  {Turning user1 into an attendee}
  $PrivilegeRevoke(user1, G1.topic, publish, webBrowse)$
  $PrivilegeRevoke(user1, G1.topic, publish, cookie)$
  $PrivilegeGrant(user1, G1.topic, publish, presenterPrivilegeRequest)$
  $PrivilegeRevoke(user1, G1.topic, publish, presenterTokenPass)$
  $PrivilegeRevoke(user1, G1.topic, publish, historyRevisit)$
  $PrivilegeGrant(user1, G1.topic, publish, downloadAck)$

  $PrivilegeGrant(user1, G1.topic, subscribe, webBrowse)$
  $PrivilegeGrant(user1, G1.topic, subscribe, cookie)$
  $PrivilegeGrant(user1, G1.topic, subscribe, presenterPrivilegeRequest)$
  $PrivilegeGrant(user1, G1.topic, subscribe, presenterTokenPass)$
  $PrivilegeGrant(user1, G1.topic, subscribe, historyRevisit)$
  $PrivilegeRevoke(user1, G1.topic, subscribe, downloadAck)$

  {Turning user2 into a presenter}
  $PrivilegeGrant(user2, G1.topic, publish, webBrowse)$
  $PrivilegeGrant(user2, G1.topic, publish, cookie)$
  $PrivilegeRevoke(user2, G1.topic, publish, presenterPrivilegeRequest)$
  $PrivilegeGrant(user2, G1.topic, publish, presenterTokenPass)$
  $PrivilegeGrant(user2, G1.topic, publish, historyRevisit)$
  $PrivilegeRevoke(user2, G1.topic, publish, downloadAck)$

  $PrivilegeRevoke(user2, G1.topic, subscribe, webBrowse)$
  $PrivilegeRevoke(user2, G1.topic, subscribe, cookie)$
  $PrivilegeGrant(user2, G1.topic, subscribe, presenterPrivilegeRequest)$
  $PrivilegeRevoke(user2, G1.topic, subscribe, presenterTokenPass)$
  $PrivilegeRevoke(user2, G1.topic, subscribe, historyRevisit)$
  $PrivilegeGrant(user2, G1.topic, subscribe, downloadAck)$
```

Algorithm 4.2 Annotation Token Pass Operation

```
Require: $user1 \neq user2$
Require: $user1 \in G1 \wedge user2 \in G1$
Require: $hasAnnotationToken(user1) \wedge \neg hasAnnotationToken(user2)$
Ensure: $\neg hasAnnotationToken(user2) \wedge hasAnnotationToken(user1)$

  $PrivilegeGrant(user1, G1.topic, publish, annotation)$
  $PrivilegeGrant(user1, G1.topic, subscribe, annotation)$

  $PrivilegeGrant(user2, G1.topic, publish, annotation)$
  $PrivilegeGrant(user2, G1.topic, subscribe, annotation)$
```

The publish phase is basically composed by the following steps:

1. The publishing phase starts when a participant wants to contribute to the session. Thus an *informative contribution* is created containing the message that this participant wants to send.

2. Still in the OCEAN client, this participant's privileges are checked in order to determine if he or she is allowed to publish such contribution type to the associated scope. If not, the system ignores this publication, avoiding unnecessary communication with the *notification service*.

3. Once allowed, the participant sends through an asynchronous RPC (Remote Procedure Call) the related contribution to the *notification service*, which is in charge of correctly delivering this message. At this point, the publisher is no more involved in the notification process.

4. The notification service starts handling the contribution through a consistency test. Such test aims at determining whether the publisher is duly registered as a participant, and whether the referred contribution is in a valid scope. If any inconsistency is detected, the system ignores the contribution, notifying the publisher through the RPC callback. Such verification is necessary for maintaining users aware of any error occurrence.

5. Thus, the notification service catalogues the contribution, associating to it a unique identifier used for global ordering. With this identification, OCEAN stores this contribution in the *co-browsing history*.

6. If the contribution contains a coordination decision, then the service updates its structures. For instance, updating participants' privileges, or creating new *co-browsing threads*.

7. After that, the service appends the contribution in the *notification shared space*, making it available for further delivering to the assigned subscribers.

8. Finally, the service uses the RPC callback to inform the publisher that the contribution was successfully processed. This callback also contains the contribution's unique identification. Hence, the publisher can maintain a local copy of a *notification shared space* subset, which works as a cache for further *history revisits*.

Completing the notification process, there is the delivery phase. Any contribution arrival at the *notification shared space* triggers this phase, which has the objective of forwarding this contribution only to the assigned subscribers. Hence, for each user properly registered in the co-browsing session, the following process is followed:

1. The *notification service* checks the participant's privileges, in order to determine if he or she is granted to receive the contribution from the associated scope. If denied, the service aborts the delivery for this participant, just marking the contribution as already treated for him or her. Especially, the *join thread* negotiation messages are considered particular communications. In this case, the service only notifies the contribution recipient.
2. If allowed, the service pushes[9] the message out to the participants. This server-push is emulated by an asynchronous RPC connection previously established by the receiver. Thus, the contribution is sent through this RPC callback.
3. After correctly transmitted, the notification service sets the contribution as already sent for this participant.
4. On the client side, the received contribution is handled in order to update management structures and present the contained message on the user interface. Moreover, the received contribution is already uniquely identified, and it is in the same way stored as a local copy of a *notification shared space* subset.

In particular, each type of received contribution has an specific handling. For instance, a *web browse* contribution causes the receiver to download and render a new web page. A received *annotation* triggers its painting over the shared content, while both *negotiation* types alert the participant with its message. The *coordination* and *session management* contributions affect the local privileges filter and make the user aware of the new session scenario. Especially, the *cookie* contribution holds HTTP cookies of the presenter and are set for the attendees with the browser extension support[10].

4.2.4 Acknowledgement Messages

Certainly, the shared web page download time is what mostly affect the continuity and the synchronism of a co-browsing thread. This is due to the fact that participants are usually geographically distributed, where computers may present different Internet connection

[9]According to Hanson & Tacy(60), server-push is a mechanism where the server pushes data out to the client without it being requested. This is often used in applications like chat, where the server needs to push messages out to its clients. This server-push mechanism can be emulated, for example, along polling techniques.

[10]This cookies setting needs browser extension support because standard browsers do not allow that an application sets or gets cookies from another domains.

capacity(61). Such heterogeneity leads to different download times for each user, inducing an asynchrony state until the user presenting the slower connection has finished. This phase denominated *asynchrony period* is detrimental to the quality of the co-browsing thread, since favors stillborn contributions. For instance, when using an audio communication tool, a presenter could need to re-explain something after all get synchronized again, because during the asynchrony period some users did still not have the discussed web page totally displayed in their client interfaces.

Thereby, it is important for a co-browsing system to maintain users aware of the asynchrony period, thus they can naturally coordinate themselves before starting contributing. Also, this awareness mechanism prevents users from often asking "*is everybody ready?*", for example. However, such facility is not usually available on most co-browsing systems, or it is not easily accessible (*e.g.* (12)).

OCEAN provides such asynchrony awareness facility as a download acknowledgement contribution, represented by the concept *download ack*, a new specialization of the *co-browse informative contribution* (Section 3.2). Every time a participant finishes downloading a web page, he or she publishes a *download ack*. This contribution has a message containing just the unique identification of the received *co-browse contribution* that has caused the web page download. When every user in the *co-browsing thread* has published their respective *download acks*, it means that the asynchrony period is over and more important, that the participants became aware of such information. Especially, at this moment, the client application could in some especial way inform the user about the end of the asynchrony period, since this information has great value itself. Additionally, setting download awareness at a participant granularity, allow participants to identify who are the slower users. Such connection quality information is useful, for instance, for allowing the presenter to request the slower attendees to look for possible problems in their private networks.

Since the *download ack* is an *informative contribution*, its usage is also coordinated by the privileges policy. At the policy's scope parameter, this contribution is affiliated with the *thread scope*, because this contribution works as an answer to the other *co-browsing* specializations, which are in the same scope. As Table 4.2 shows, in the function parameter it is defined that attendees can only send *download acks*, while the presenter can only receive them.

Note that, with this privilege's configuration, only the presenter has access to the asynchrony awareness information. We could have set all privileges as granted, allowing all group participants to become aware of such download acknowledgement information. However we have taken this design decision in an effort to save the notification service from a contribution flooding (mainly considering great thread groups). Table 4.3 shows the number of exchanged messages in both approaches.

Table 4.3: Comparative of Download Ack Distribution Approaches: considering n the number of thread group members, this table presents how many accesses to the notification service are necessary for providing asynchrony awareness, on each browsing action.

Distribution Approach	**Sent Acks**	**Received Acks**	**SUM**
All Participants	n	$n^2 - n$	n^2
Only Presenter	$n - 1$	$n - 1$	$2n - 2$
Saving	1	$n - 1$	n

Even if asynchrony awareness is an important facility for any participant, the presenter is certainly the better candidate to receive this information, since he is the one who "guides" the browsing activity and might probably have the control of the annotation and other external communication tools. In further developments of OCEAN, we intend to evolve this facility in order to provide synchronism awareness for attendees too. For instance, it would be possible to provide such awareness in a higher granularity, informing only when the asynchrony period is over, leaving the participants granularity only for the presenter.

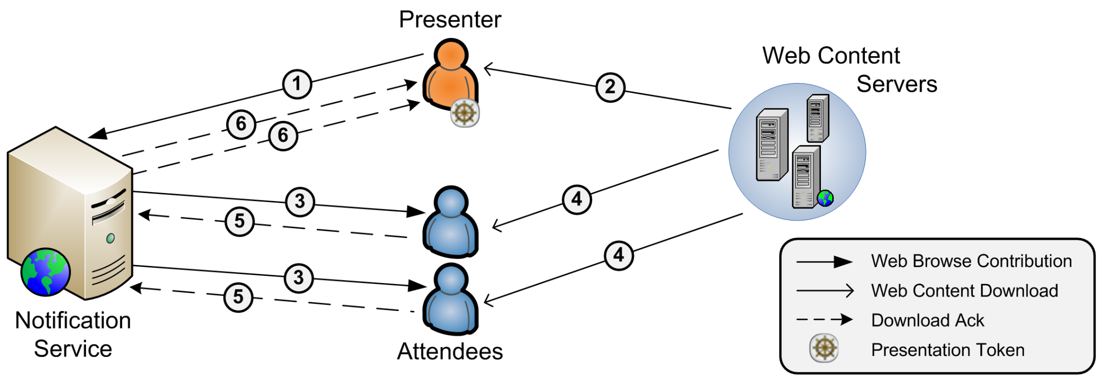


Figure 4.3: URL Broadcasting Scenario

Illustrating the asynchrony awareness facility, Figure 4.3 presents a thread group with three members, where the presenter is publishing a *web browse* contribution to the attendees. The arrows represent the network interactions made in this scenario, and following their labels, the interaction process is detailed:

1. The presenter wants to navigate and publish a new URL to the notification service;
2. The presenter simultaneously starts to download the desired content directly from the web content server;

3. After processing the published contribution (containing the URL), the notification service notifies all the attendees;
4. After receiving the notification, each attendee downloads the web content from the aforementioned web content server;
5. As soon as each attendee has finished the content download, the client acknowledge the notification service that it is synchronized;
6. Finally, this service individually redirects the acknowledgment messages from attendees to the leader, as they reach the server, thus keeping him/her aware of the attendees' synchronization state.

It is important to understand why only co-browsing actions are acknowledged. According to OCEAN design all other types of contribution do not demand great amount of data or external system access (e.g.: external web servers), thus not generating great asynchrony periods. An annotation contribution, for example, can be compared to a message in an ordinary chat tool, which simply do not notify the writers about every message correctly delivered. However, it would be interesting an NACK awareness facility. Such a feature consists of the contribution receiver notifying back the publisher on errors occurrence. Actually, in OCEAN, the notification service only handles errors inside a specific notification process phase. So, when an error occurs on the receiving phase, the related publisher is not warned. However, we also intend to design a NACK awareness facility as soon as possible.

4.3 Conclusions

OCEAN has been designed in terms of the distributed architecture and the notification protocol, both aiming to provide performance and flexibility for co-browsing sessions. Comparing with the most adopted proxy-based approach, OCEAN's architecture was conceived in an effort to avoid bottlenecks on the server-side, taking advantage of each participant's bandwidth, in order to promote a better synchronous collaborative experience for distributed users.

Considering the notification protocol, it extends a well-founded communication paradigm (publish/subscribe), essentially aggregating flexibility and transparency through the privileges policy. Amongst the inherited characteristics, certainly the most relevant is the loosely-coupling between producers and consumers (58), which favors the management of dynamic co-browsing sessions and threads. Also, the system became easily configurable by privileges changing, and extensible by easily support the adding of new contribution types. Furthermore, according to

Dyck et al.(62), event-driven protocols based on TCP (Transmission Control Protocol), likewise the notification protocol, are relatively simple to implement, and perform well when events are rare and guaranteed delivery is required, both characteristics of our supported contributions.

In fact, the design of OCEAN presented in this chapter has already extended the conceptualization formalized in the previous chapter. Firstly introducing the notification protocol concepts (Figure 4.2) and in the sequence, introducing specializations of the *informative contribution* concept. These latter are summarized in Figure 4.4.

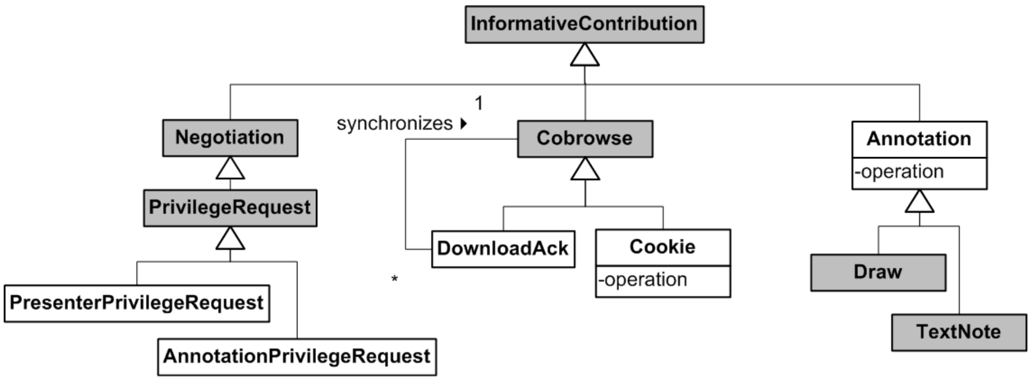


Figure 4.4: Design Introduced Concepts: concepts' specializations introduced by the OCEAN design. The concepts *in gray* composes the OCEAN conceptualization (Chapter 3), while the others *in white*, have been introduced by this chapter.

Describing the concepts introduced, *presenter privilege request* and *annotation privilege request* (Section 4.2.2) remain to negotiation primitives of presentation and annotation tokens respectively. In the sequence, the *download ack* (Section 4.2.4) fosters a synchronism awareness facility, while *cookie* (Section 4.1.2) is useful for sharing HTTP cookies. Especially, this *cookie* concept has an *operation* attribute, used by the OCEAN client application (more precisely, the browser extension) to correctly handle such contribution. Its possible values are: "*set*", for setting a new HTTP in the browser, and "*del*", for removing a cookie previously shared by this mechanism. At last, besides defined at the OCEAN conceptualization, the *annotation* concept is modified at the design level, being augmented with the attribute *operation*. Similarly in *cookies*, this attribute is used as input for the application client procedures. In case of annotations, the possible *operation* values are: "*create*", defines a new painted object; "*change*", update an existing object, for instance moving in for another position; and "*remove*", that erases a previously painted annotation from the user interface.

We are aware that the proposed design could also present dependability problems, for instance single failure points and some performance bottlenecks. This is mainly due to its centralized entities: the application server, and also the *notification shared space* structure. In fact, some of these might be minimized depending on how these elements would be implemented. In this regard, Chapter 5 describes an implemented prototype of OCEAN.

5 The Implemented Prototype

Aiming at demonstrating the feasibility of our proposal, a proof-of-concept software prototype for OCEAN has been developed. This implementation was built on top of the GWT (Google Web Toolkit), an open source framework for web applications development (63). GWT provides a number of features, such as (*i*) cross-browser JavaScript compiler, allowing developers to program the client-side in Java; (*ii*) a set of reusable user interface widgets; and also (*iii*) a simple and powerful RPC framework (GWT-RPC) for communicating client and server, which is used for supporting the *Notification Protocol*.

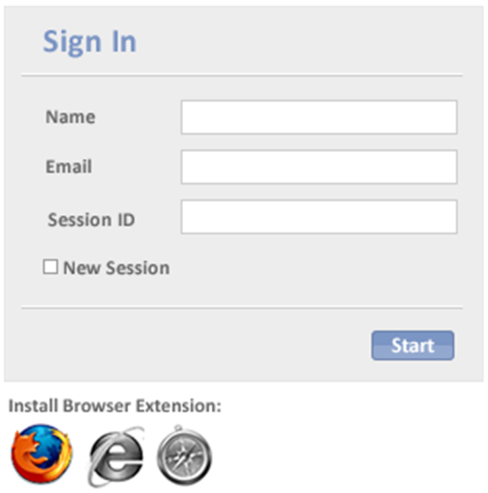


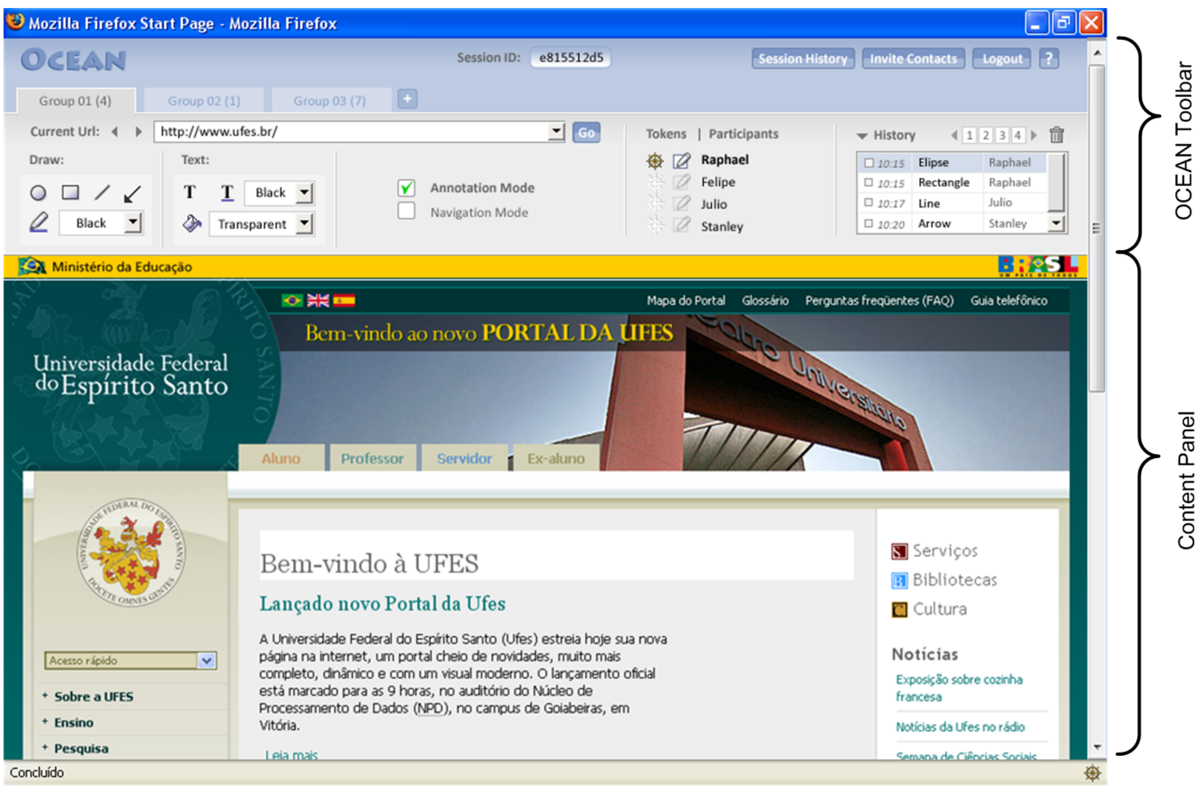
Figure 5.1: Login Screen: this form allows a user to create or join co-browsing sessions.

In this chapter, we describe the developed prototype. We start by showing the two main ways that a user can get in a co-browsing session. The user can start from scratch, creating a new session, or he can join an existing one, informing the respective *session id* during the login process. Both features are supported by the login screen, depicted in Figure 5.1. Besides providing login facilities, this screen also provides links for downloading and installing respective browser extensions[1]. It is important to mention that, at this prototype version we are not interested on users' account management and information security issues. Due to

[1]Actually we have only developed an extension for Mozilla Firefox browser.

that, the login screen does not provide any authentication mechanism, for instance, a password verification.

Right after a successful user registration, a *login informative contribution* is published (see Section 4.2.2) and the main co-browsing window (illustrated in Figure 5.2) is presented. From this window the user can access all the main features that he/she can perform in a co-browsing session. As mentioned on the previous chapter, OCEAN client application consists of a dynamic web page served by the application server. This page's frontend is thus the main co-browsing window which is divided into two areas . On the top part, there is the *OCEAN Toolbar*, which contains a set of user interaction panels, detailed further in this chapter. In the middle of the co-browsing window, there is the *Content Panel*, where the shared web contents are presented, through an enclosed inline frame[2].



Session ID: e815512d5

Current Url: http://www.ufes.br/

[✓ Annotation Mode] Navigation Mode

10:15	Elipse	Raphael
10:15	Rectangle	Raphael
10:17	Line	Julio
10:20	Arrow	Stanley

Bem-vindo à UFES

Lançado novo Portal da Ufes

A Universidade Federal do Espírito Santo (Ufes) estreia hoje sua nova página na internet, um portal cheio de novidades, muito mais completo, dinâmico e com um visual moderno. O lançamento oficial está marcado para as 9 horas, no auditório do Núcleo de Processamento de Dados (NPD), no campus de Goiabeiras, em Vitória.

Figure 5.2: User Interface Overview: this is the OCEAN client application GUI, used for participating in an ongoing co-browsing session.

We present, in Section 5.1, a detailed description of all interaction panels, explaining how the designed mechanisms for contributing in the sessions are supported by them. Section 5.2 describes the implemented Mozilla Firefox browser extension, designed to allow our application to access shared web pages information. We finalize this chapter, in Section 5.3,

[2]Inline frames, defined by the HTML tag *iframe*, makes it possible to embed an HTML document inside another HTML document. <Wikipedia web site: en.wikipedia.org/wiki/IFrame>

discussing some proposal restrictions and specializations adopted in this implementation, and also, outlining some conclusions and future work.

5.1 Interaction Mechanisms

The OCEAN *Toolbar* (Figure 5.2) gathers the data and mechanisms for enabling users to participate in a co-browsing session. This toolbar is composed by some interaction panels, specialized for each possible user activity. These panels are visually organized in accordance with the acting scope[3] of their related contributions, that means for which scope (*i.e.*: *session scope* or *thread scope*) this contribution has granted privileges. Such visual organization was assumed in an effort to provide users with little distinction about where they are participating. This way, the *Toolbar* is itself divided into two scope related sections, namely the *Session Scope Toolbar* on the top, and the *Thread Scope Toolbar* next on the bottom .

5.1.1 Session Scope Toolbar

The *Session Scope Toolbar* is focused in Figure 5.3, and holds the interaction mechanisms used for publishing contributions in the whole session scope. In the figure, we can notice the co-browsing session identifier at the center of toolbar's first line. This identifier is generated by the *Session Manager* at the session creation, in order to uniquely identify a session in the whole system. It is used as a key allowing other participants to join this session, through the Login Screen. In the sequence, there are the *Session History*, *Invite Contacts*, *Logout* and *Help* buttons . These controls trigger respectively the activities of accessing the co-browsing history, inviting participants to join the same co-browsing thread (*group join invitation*, see Section 4.2), leaving the co-browsing session (*logout*) and accessing help information about OCEAN usage.

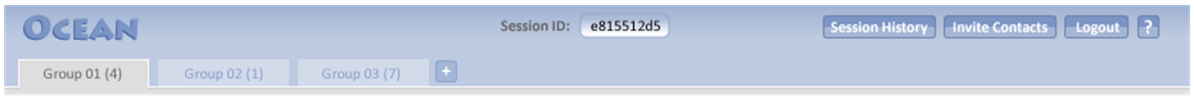
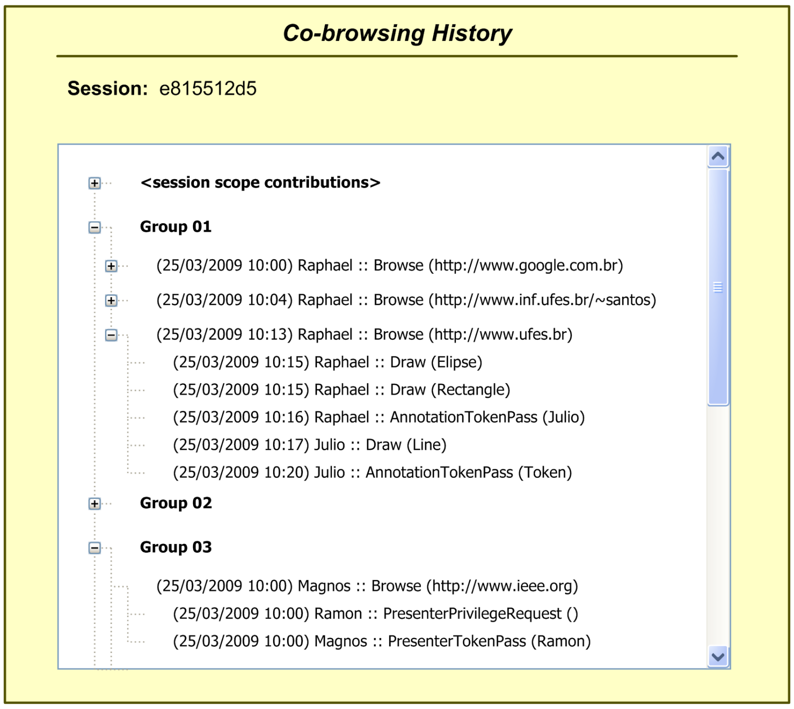


Figure 5.3: Session Scope Toolbar

In particular, the *Session History* triggers the opening of a popup panel containing the history of the whole session. Actually, such history is presented through a 3-level tree structure, as presented in Figure 5.4. In this tree, the highest level represents the different scopes. In other words, the session scope and each specific group. In the second level, the checkpoint

[3]A parameter used for determine the distribution range of an *informative contribution*. More details in Section 4.2.2

co-browsing resource participations are organized (see Section 3.2), for instance, navigation contributions. Finally, the remaining general contributions are the leaves. In particular, *download ack* messages are omitted in this tree in order to not overload this view with such pure awareness information.


Co-browsing History

Session: e815512d5

<session scope contributions>

Group 01

(25/03/2009 10:00) Raphael :: Browse (http://www.google.com.br)
(25/03/2009 10:04) Raphael :: Browse (http://www.inf.ufes.br/~santos)
(25/03/2009 10:13) Raphael :: Browse (http://www.ufes.br)
(25/03/2009 10:15) Raphael :: Draw (Elipse)
(25/03/2009 10:15) Raphael :: Draw (Rectangle)
(25/03/2009 10:16) Raphael :: AnnotationTokenPass (Julio)
(25/03/2009 10:17) Julio :: Draw (Line)
(25/03/2009 10:20) Julio :: AnnotationTokenPass (Token)

Group 02

Group 03

(25/03/2009 10:00) Magnos :: Browse (http://www.ieee.org)
(25/03/2009 10:00) Ramon :: PresenterPrivilegeRequest ()
(25/03/2009 10:00) Magnos :: PresenterTokenPass (Ramon)

Figure 5.4: Co-browsing Session History

Still in the *Session Scope Toolbar* (Figure 5.3), the group coordination primitives are available through the tab panel controls, located in the toolbar's second line. In this approach, the selected tab stands for the thread group that the user is member at a given time. Whereas, the other unselected tabs stand for the remaining thread groups, available in the session. Thus, the act of selecting a different tab is interpreted as: the users is leaving the current thread group (*leave thread* primitive) and joining the thread group related to the selected tab (*join thread* primitive). This situation has been illustrated by Figure 3.2. At last, the *create thread* primitive is accessible through the "plus" button, located next the last tab.

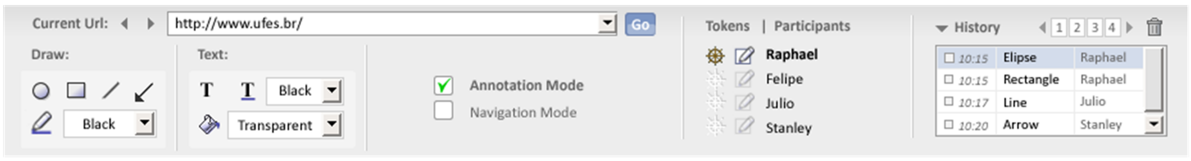
This tab-based implementation approach was chosen due to two main reasons. Firstly, there was an effort to provide an environment as most similar to a standard browser as possible. These standard browsers usually implements a browser tab facility[4] aiming at offering for their users a nice way for organizing their parallel navigation contexts. In doing so, we hope that

[4]The Mozilla Firefox web site (www.mozilla.com/en-US/firefox/features/#tabs) argues that this browser tabs facility, at first glance, look like little labels living above the site users are currently visiting. But they are a brilliant way to browse multiple sites at once. Simple and easy, users can think of these tabs as the electronic version of a well kept filing cabinet, with the tabs as the dividers and the sites as the content kept in folders. Each new site appears as a new tab (not a new window) and can be accessed in one click.

OCEAN users feel more comfortable when using our application. The second reason is related to providing an easier mechanism for coordinating users while they are participating in different co-browsing threads. Once a user has granted a privilege to join a thread group, any extra authorization request for performing this task is unnecessary. Hence, users can easily join a thread and check out what other groups are doing, without being annoyed with authorization requests and responses. As a remark, CoLab (11), the OCEAN's most similar co-browsing system, is based on authorization messages for coordinating such groups. However, the CoLab coordination mechanism focuses on social commitments between participants pairs, having groups just as a consequence of such commitments (see Section 3.1.1).

5.1.2 Thread Scope Toolbar

The Thread Scope Toolbar (Figure 5.5) gathers the interaction panels that are tightly related to the specific co-browsing thread the user is participation. Certainly, the most important is the *Address Bar*, that is illustrated in Figure 5.6. Similar to any standard browser, this panel allows the presenter to input the URL he wants to navigate to. Also, it displays to all participants the URL of the current shared web page, in order to promote awareness. Moreover, this component is a combo box[5], and stores in its drop-down list, a history of all browsed URLs in this thread. Such history enables the presenter to easily revisit any co-browsed page.



10:15	Elipse	Raphael
10:15	Rectangle	Raphael
10:17	Line	Julio
10:20	Arrow	Stanley

Figure 5.5: Thread Scope Toolbar

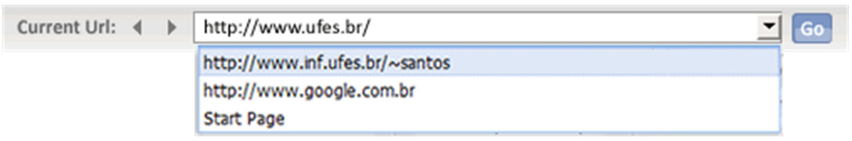
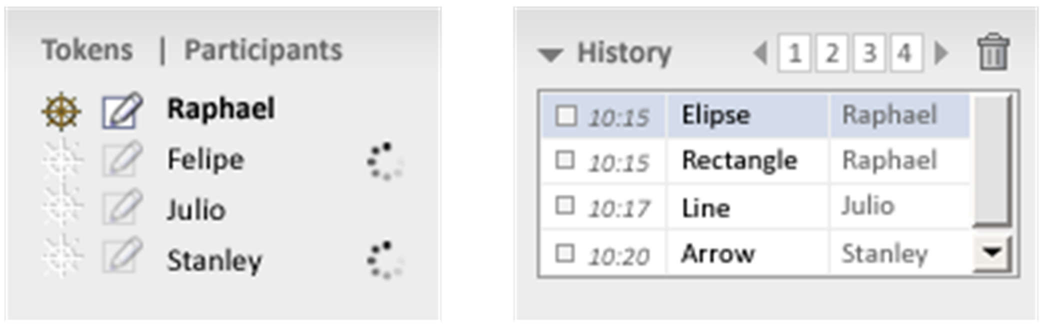


Figure 5.6: Address Bar

With respect to the co-browsing history, this feature is important for allowing users to review any generated information during the session, not only the browsed URLs. In order to facilitate the access to this feature, a subset of the co-browsing history, containing the

[5]A combo box is a commonly-used graphical user interface widget. It is a combination of a drop-down list or list box and a single-line textbox, allowing the user to either type a value directly into the control or choose from the list of existing options. An example of this use is the address bar of graphical web browsers. <Wikipedia web site: en.wikipedia.org/wiki/Combobox>

contributions published in the co-browsing thread is directly available in the *History Panel* (Figure 5.7(b)). This panel is always visible on the interface, not depending on any secondary screen (as it is the case for the complete session history). This component is important to provide awareness concerning users' contributions to the hole group, for instance informing on the fly, the type of an annotation, when it was painted and by who. In addition to that, the *History Panel* also provides a shortcut for erasing annotations objects.


Tokens | Participants

Raphael
Felipe
Julio
Stanley

History 1 2 3 4

☐ 10:15	Elipse	Raphael
☐ 10:15	Rectangle	Raphael
☐ 10:17	Line	Julio
☐ 10:20	Arrow	Stanley

(a) Members List (b) Contributions History

Figure 5.7: Thread Awareness Panel

Figure 5.7(a) presents the *Users Panel*, which is responsible for showing the group members and their current privileges depicted with token icons. Besides, this panel allows token holders to pass them to other users (with or without previous request), and also, allows attendees to request tokens. At last, it also shows a loading icon informing the presenter about everyone's synchronization state (see Section 4.2.4).

At last, the Thread Scope Toolbar contains interaction panels related to the annotation feature. Annotations provide the capability of highlighting some specific shared content area. This feature is very useful, for instance, to synchronously drive participants' attention on parts of the presented content. OCEAN supports annotations using comments, ellipses, rectangles, arrows and lines, with different colors and backgrounds. In particular, draw elements are rendered using small colored points, geometrically organized (64), while the comment notes are made using GWT text boxes (63). Thus, the annotation token holder can point, mark, highlight, and comment passages of the presented content, easily choosing a tool in the *Annotation Panel* (illustrated in Figure 5.8).

An annotation scenario is depicted in Figure 5.9, where strokes and comments were added over the shared content (URL: www.ufes.br). Moreover, all annotation objects are removable and movable (drag and drop) by the annotation token holder, and these actions are reflected on all attendees. In other words, all attendees view an annotation element with the same color, size and position as added by the annotation token holder.

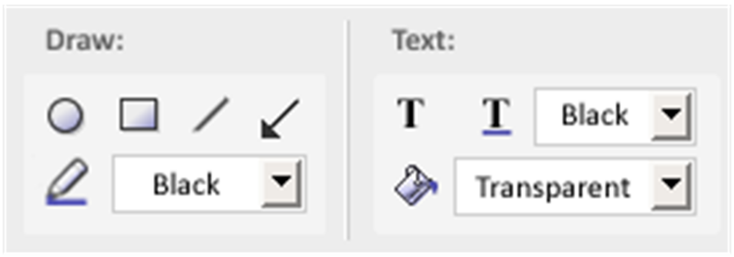
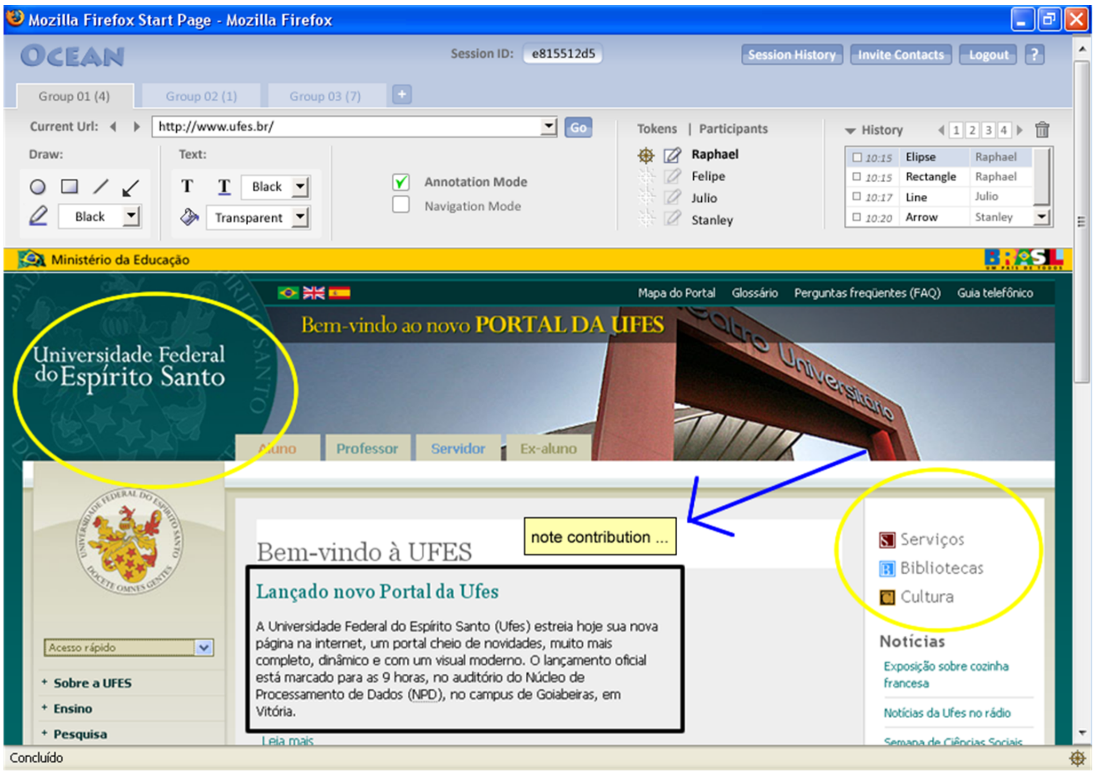


Figure 5.8: Annotations Tools



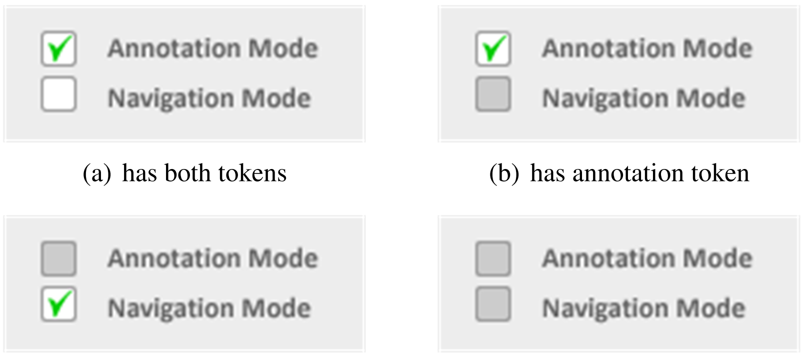
Lançado novo Portal da Ufes

A Universidade Federal do Espírito Santo (Ufes) estreia hoje sua nova página na internet, um portal cheio de novidades, muito mais completo, dinâmico e com um visual moderno. O lançamento oficial está marcado para as 9 horas, no auditório do Núcleo de Processamento de Dados (NPD), no campus de Goiabeiras, em Vitória.

Figure 5.9: Some Annotated Objects

Still regarding annotations, the toolbar presents a panel implementing the *Interaction Mode Switcher*. This panel (Figure 5.10) is useful for determining the semantic of clicks over the shared content. For example, when the same user is the presenter and also holds the annotation token, he or she can use this panel for choosing between drawing or navigating. So, when drawing any stroke, the user does not need to be concerned with clicking on shared web pages hyperlinks, accidentally moving the whole group to another web site. Therefore, this panel specifies whether the user is interacting directly on the shared web page or on a "transparent" annotations board placed over this web page.

Such layered distinction between the shared web page and a transparent annotations board overlaying it provides a simple management of these elements. For instance, we do not need to modify the web page by painting the annotation. Actually, when an annotation is created, it



(a) has both tokens (b) has annotation token

(c) has presentation token (d) has no tokens

Figure 5.10: Interaction Mode Switcher: possible interaction modes considering annotation and presentation privileges

is just presented on the overlaying transparent board, and after that, OCEAN notifies the others with the annotation's coordinates, type and color. Thus, once this notification is received, each attendee's client just need to re-paint on its respective annotations board an identical annotation on the informed coordinates. This annotation synchronization mechanism allows OCEAN to save network and server resources, since it only sends a small description of the stroke, instead of the full painted image . Indeed, such (x,y) coordinates mechanism has also been adopted as an efficient message encoding for whiteboard systems, for instance in (62).

However, this annotation's transmission approach could not work properly when the creator and the receiver are using different screen resolution and/or browser window sizes. For instance, the user *S* intends to highlight an image by using a circle. In this matter, he creates such annotation object, and his client application sends its coordinates to the thread group members. This way, the user *R* receives this contribution, and re-paints it. But, user *R* has opened his browser in a small size, and due to that, the image that should be highlighted was rendered elsewhere, and the re-painted circle is highlighting a wrong content. In an effort to solve this problem, without losing transmission performance, we have extended the OCEAN conceptualization and *Notification Protocol*, adding a new information contribution type: *content width broadcast*. At the session creation, the client application observes the width (in pixels) of the session creator Content Panel. After that, this client automatically publishes this information to the session scope. Thus, for every participant that logs in this session, this contribution will be received, and the respective client will set its content panel with this received fixed width. This way, if a participant uses a minor resolution or browser window size, his browser will automatically present a scroll bar, but the content rendering positioning will be not affected.

5.2 Browser Extension

In accordance with the OCEAN architectural design, the system relies on a browser extension in order to allow OCEAN's scripts to access relevant information from the shared content, when such content comes from a different domain. Due to the vast documentation support and available open source extensions, we have chosen to implement an extension for the Mozilla Firefox standard browser.

This implemented extension is essentially composed by a set of scripts running in background, not demanding any user interference. Additionally, the only provided user interface object is an icon, that appears on the browser's status bar when a co-browsing session is active. So, the extension is meant to accomplish two main tasks: observing events and managing shared cookies.

Events Observer: This task is the most important requirement for the extension, since the purpose of its development is accessing shared content information. In the current implementation , we have focused on observing when a user clicks on a hyperlink. Such event must be observed in order to notify the group attendees about the new shared web site.

Also, these navigation events are observed on attendees' browsers. However, in this case, these events are canceled before browser following the clicked hyperlink. This is assumed because the attendees do not have navigation privileges.

Shared Cookies Management: This is an important feature that allows the collaboration session to share cookies created by web sites. Cookies are usually used for maintaining authenticated user's session and storing user's preferences. Thus, OCEAN allows its users to have the same view of the web site.

At the presenter's side, the extension observes any new received cookie from the shared content domain, and notifies the OCEAN client application, which publishes a *cookie* (with a "*set*" operation) contribution. In the same way, when the shared content removes these cookies or the presenter navigates to another domain, the client application publishes another *cookie* (with a "*del*" operation) contributions.

Conversely, at the attendees point of view, when one of these cookie contributions are received, the respective action is requested to the browser extension. After accomplished, the current URL is reloaded in order to download information regarding the same state.

5.3 Conclusions

This chapter described the prototype implementation as a proof-of-concept that allows users to collaborate in a co-browsing session through a standard browser. The developed software covers most of the requirements presented in the specification and design of OCEAN and for that reason, we consider that it reaches its proposed objectives.

In the current version of this prototype, some features have been restricted either because they would demand too much development effort or because they would not be really relevant when evaluating the application behavior . For instance, making co-browsing session histories available when the session is over. We are aware about the importance of this feature, but it is more related to a knowledge repositories management than to the co-browsing collaboration sessions management itself. Due to that, we have classified this feature as an improvement to be developed as future work.

Considering all implemented features, we recognize that there are many points to be improved and we present some points hereafter. If we want to transform it in a public usable co-browsing suite, users accounts management and a more flexible content panel (*e.g.*: with more negotiation mechanisms than just token requests or session invitations) must be provided. Another important improvement is the management of group tabs. In this matter, it is important to design how the group tab panel should behave with a great number of groups in the session. We intend, in the future, to develop some organization based, for instance, on group's contribution rate or user interest matching. This way, only the most relevant groups would be presented on tabs, and other ones would be available in a secondary list container. Additionally, it would be interesting to add some group context awareness for non members, for example, using the shared web page title as the tab label (as do standard browsers), or even implementing any mechanism allowing to indicate group activity/inactivity.

Regarding the browser extension, it would be valuable to develop extensions for others browsers, for instance, Microsoft Internet Explorer and Apple Safari, aiming at reaching a greater number of users. Even though Mozilla Firefox has been more adopted lately, according to some statistics presented in (65) and (66). Also in the browser extension, there are many desired improvements. Amongst them, the most relevant is to extend the number of observed navigation events. For instance, form-filling or scrolling. For such task, we intend to investigate browser macro recording tools as in (67)(68). Such tools are capable of capturing a large number of different user interactions on a web site , not only a URL browsing. These interactions are registered in a simple macro language, in order to be further reproduced or personalized. Using such technology, we intend to increase co-browsing capabilities, for instance, enabling users to co-browse on dynamic web sites, that uses AJAX[6] components for example.

In summary, this software prototype is the direct result of our proposal. Its data models are implemented exactly following the ontology-based conceptual models (Chapter 3), and based on our development experience, we could experiment that this *domain modeling* was an effective method for enabling the rapid prototyping of applications (see Section 3.2.1). Whereas its distributed architecture, service logic and business rules follow design level specific definitions (Chapter 4) , which have favor a lot our development effort through the use of the same standardized *notification protocol* as the basis of all prototype procedures.

Last, we see the resulting prototype as a potential good alternative for the revised related work, providing innovative features on the co-browsing paradigm. In order to a preliminary evaluation of our proposal, based on this specific implementation, the next chapter evaluates performance aspects of OCEAN.

[6]an acronym for *Asynchronous JavaScript And XML* that represents a group of interrelated web development techniques used to create interactive web applications or rich Internet applications. <Wikipedia: en.wikipedia.org/wiki/Ajax_(programming)>

6 Performance Evaluation

This chapter describes the performance evaluation of OCEAN. The main objective is to quantify some measures of interest relevant to our co-browsing service using the implemented prototype as a testbed for experiments. In order to provide a platform that allows users to participate in collaboration sessions, a collaborative system has a special requirement of responsiveness for supporting suitable user interaction. Thus, given that these systems are essentially interactive, the interaction *response time* is a key metric.

In fact, according to Dyck et al.(62) such metric is an important characteristic that differs synchronous groupwares from other network-based applications. In particular, these authors argue that the performance of synchronous groupware is typically measured in terms of *feedback* and *feedthrough* times, and are characterized by workloads involving frequent small bursts of information. *Feedback* and *feedthrough* are concepts related to *response time*, being respectively defined as the time spent from a user performing an action and himself seeing the results of that action, and the time from a user performing an action to other users seeing its consequences (62). In particular, we focus on the OCEAN's responsiveness in general, either in terms of feedback, feedthrough or the whole time needed for users to become synchronized.

In this regard, we have focused on evaluating two specific characteristics of OCEAN, both affecting directly the responsiveness perceived by users and clearly the quality of the service. On one hand, there is an *delay* imposed by the notification protocol . If this delay were too long, then it could be better to navigate individually instead of using our collaborative system. This issue is widely discussed in section 6.1. On the other hand, as the number of users increases, it is crucial to evaluate OCEAN is system capacity. In our approach, the application server has the role of mediating every communication, so there is a risk of becoming a bottleneck for the whole system. We are specifically interested on the *scalability* of the application server, i.e., to determine the workload the server supports without degrading the quality of service of co-browsing sessions. Such issue is studied in section 6.2.

6.1 Notification Protocol Delay

Introduced by section 4.2, the *notification protocol* supports and manages every users' contributions in OCEAN. Its goal is to act basically as a transparent application-level multicast protocol, automatically publishing participants' contributions for the appropriate destinations. Thus, it is very important that this protocol does not impose long delays affecting the whole service responsiveness. In particular, if our protocol delays to deliver a *web browse* contribution, then the relative *asynchrony period*[1] will be too long, leading the co-browsing thread to be too slow. Since co-browsing interactions obviously are the most relevant interactions in a co-browsing system, the *web browse* and the respective *download ack* contributions are the main focus of our analysis in this chapter.

In an effort to demonstrate that the proposed *notification protocol* achieves OCEAN's transmission requirements, we have performed a set of experiments intending to determine the extra cost of using this protocol for collaborative web browsing. These observed costs were compared with a traditional single mode browsing activity, aiming at understanding how significant they are. Thus, the *notification protocol delay* can be defined as the extra time required for transforming a traditional single browsing action into a co-browsing action, through the *notification protocol*. In other words, having the time a user spent downloading a web page through a standard browser, how long does it take for OCEAN to allowing a group of users to synchronously browse the same web page.

Given a co-browsing thread (Section 3.2.2), the *asynchrony period* starts when the presenter publishes a web browse contribution, and finishes when he/she receives the last download acknowledgement. For this reason, the delay imposed by the protocol can be measured by the difference between the *asynchrony period* and the web page *download time*. It is interesting to note that the *download time*[2] is a local measure obtained in all thread group members. It represents the time taken to download the web page in a traditional solo browsing paradigm. Figure 6.1 illustrates the relation among the *asynchrony period*, the *notification protocol delay* and web page *download times*, through an example of a thread group with three members[3].

In the proposed architecture, clients download web pages by their own, accessing directly the respective target web servers not going through the co-browsing server to retrieve the web

[1]The time taken to get all participants synchronized in the same shared web page. See section 4.2.4.

[2]The *download time* measures the entire web page download, for example, including the time spent downloading the HTML file and its embedded images. This period starts by the browsing request, and finishes when the inline frame (OCEAN Content Panel) triggers a *load* event. More details about this event can be found at W3Schools.com <www.w3schools.com>

[3]This example follows the same scenario presented by Figure 4.3, however in a clients perspective of the asynchrony period.

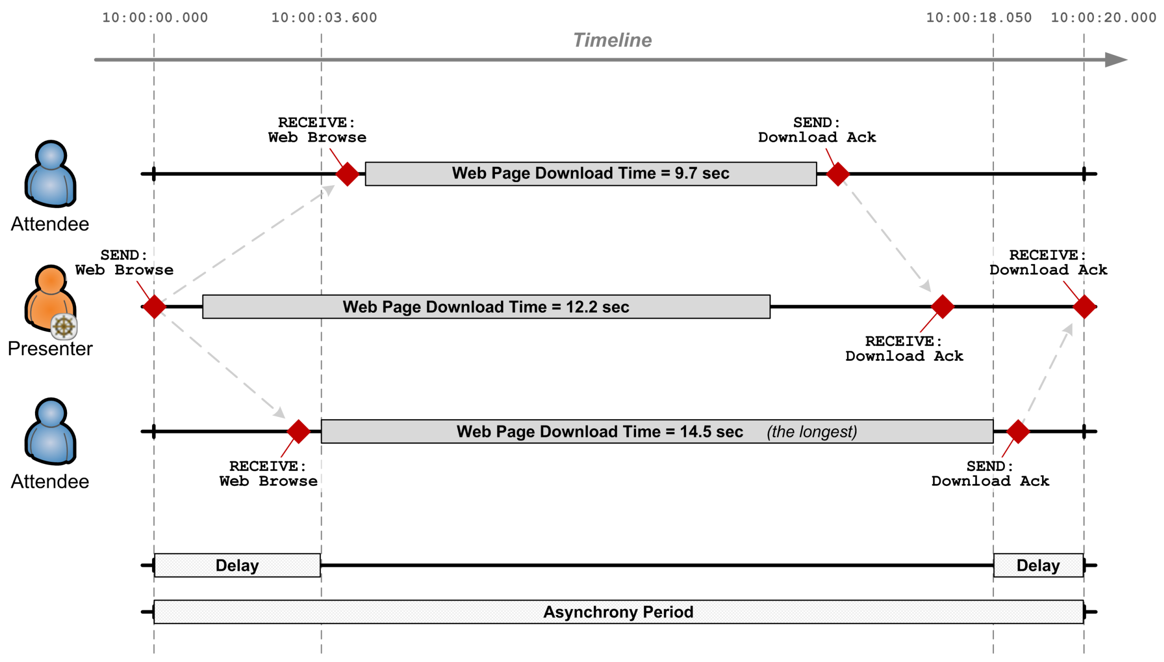


Figure 6.1: Notification Protocol Delay: an out of scale illustrative example of how the notification protocol delay measure is obtained, from the observation of events occurred during the asynchrony period of a co-browsing thread.

page. Clearly, their download times depend on their network connection as in a single mode browsing. Thus, to compute the delay of the protocol, we consider the *longest download time* of the co-browsing cycle[4], which is experienced by the slowest client. Note that this client has probably the narrowest network bandwidth, or packets have experienced long queues or even retransmissions because of network congestion along the path.

Therefore, the formal way of obtaining the notification protocol delay is defined in Equation 6.1, which considers a thread group of size n, where the presenter (*pres*) sends web browse contributions (*browse*) and receives respective download acks (*ack*) from each member i of this group.

$$delay = \overbrace{max_{i,n}\left(timestamp_{rcv,i,ack}\right) - timestamp_{snd,pres,browse}}^{asynchrony\ period} - \overbrace{max_{i,n}\left(download_i\right)}^{longest\ download\ time} \tag{6.1}$$

This assumption of considering the *longest download time*, is justified by the fact that if a group of people decided to collaboratively browse, as a consequence they are willing to wait at least the slower participant. Therefore, that protocol delay measure captures only the extra time imposed by the *notification protocol*, not counting time periods inherent to the proper

[4]Co-browsing cycle is the sequence of interactions that happen in an asynchrony period, depicted by figure 4.3

collaboration paradigm.

In order to quantify such a measure, determining its order of magnitude in comparison with the whole asynchrony period, we have performed several experiments with the implemented prototype. The adopted evaluation methodology consists of obtaining quantitative measures of prototype's components as co-browsing sessions proceed, varying the following parameters: (*i*) the number of online users (organized in the same thread group); (*ii*) the set of co-browsed web pages; and finally (*iii*) the environment, which comprises experiments in a laboratory (dense and few distributed scenario) and comparing to field experiments (sparse and well distributed scenario), where users are distributed over distinct Internet locations (16).

Note that in the Internet scenario, the messages of the system can be mainly delayed by the following components: (*i*) an inherent propagation (physical distance from participants to OCEAN's application server) (*ii*) a packet transmission (depending on users' bandwidth and capacity of the end-to-end path) and (*iii*) a network congestion (long queues at the routers).

Therefore, using this evaluation methodology we aim to demonstrate that the *notification protocol delay* is quite acceptable for a co-browsing paradigm, not degrading the quality of service. Also, a second objective is to observe that this *delay* variation tends to be smooth, either as the number of thread group members increases or the co-browsed web pages' content sizes vary, showing the effect of these parameters. So, the obtained results are subsequently presented, focusing on specific parameters effect over *delay* measures on laboratory and Internet scenarios.

6.1.1 Number of Users Effect

Trying to discover if the notification protocol delay is significantly sensible to the number of users, a set of experiments were conducted varying the number of participants from 2 to 22 using the OCEAN prototype in one co-browsing thread. Each of these experiments was composed of co-browsing having 10 web sites located at distinct domains (e.g. ".org", ".com").

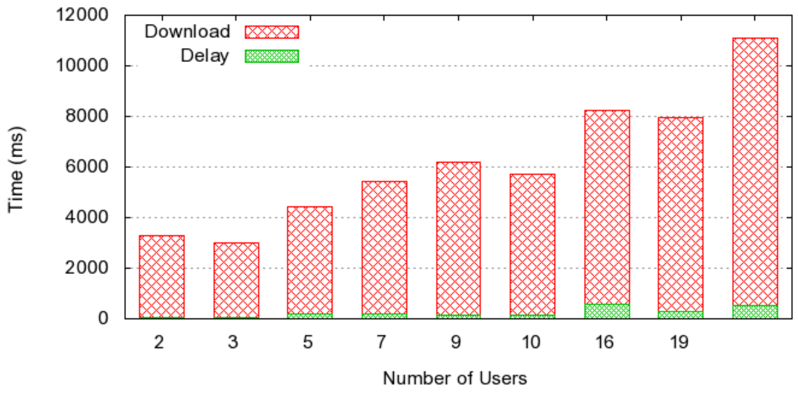
A first bulk of experiments was conducted in a local network of $10Mbps$ connecting all the participants to the OCEAN application server, and a link of $5Mbps$[5] connecting this LAN (Local Area Network) to the Internet, where are the contents' web servers.

Figure 6.2(a) presents a proportional comparison between the observed downloading and delay. In this scenario, the delay had an average of $262 \pm 219msec$. Although we can observe a variability of the delay, it tends to be proportionally lower as the number of users increases

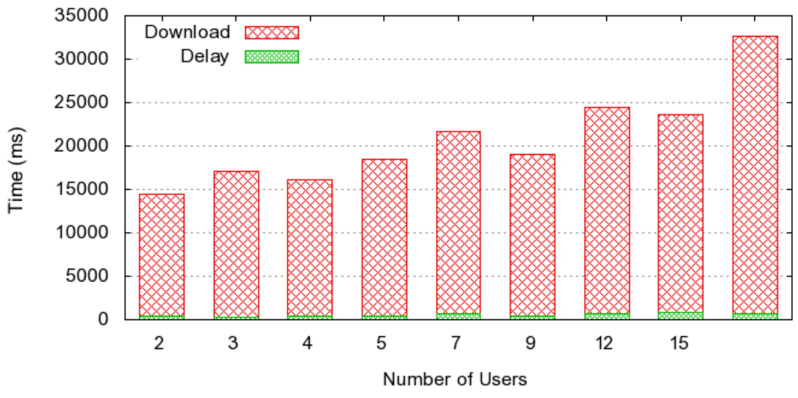
[5]This link was under a rate limit policy, due to network administration.

especially if compared to the whole asynchrony period.

Following the methodology, a second bulk of experiments was conducted in an heterogeneous Internet scenario, with users distributed through worldwide locations (e.g. Brazil, USA and Germany). In this scenario, the users' Internet connection bandwidth varies from $100Kbps$ to $10Mbps$. Additionally, the user's network path to the OCEAN server has around $15 \pm 10 hops$, with an average network delay to the OCEAN server of $230 \pm 150 msec$[6]. Figure 6.2(b) depicts the average results for this distributed scenario, in which the notification protocol delay had an average value of $559 \pm 163 msec$. Additionally, the observed proportional values of the notification protocol delay are presented in Table 6.1;



(a) LAN - *from: 09/10/2008 08:40 PM , to: 09/11/2008 01:03 AM*



(b) Internet - *from: 09/13/2008 06:34 PM , to: 09/13/2008 10:20 PM*

Figure 6.2: *Proportional Delay - Number of Users Parameter*: a report of medium *notification protocol delay* and *longest download times* observed in experiments, that have focused on the effect of number of users raising over such measures.

As expected, there is a significant increase on the asynchrony period from the laboratory to the Internet scenario, besides the conducted experiments for both scenarios have been based on the same set of web sites. For instance, taking 9 users, we observe $19,014 msec$ (Internet)

[6]The network delay values are estimated using the command *ping*.

Table 6.1: Experiments Data - Number of User Parameter: the observed notification protocol delay values proportional to the total asynchrony period.

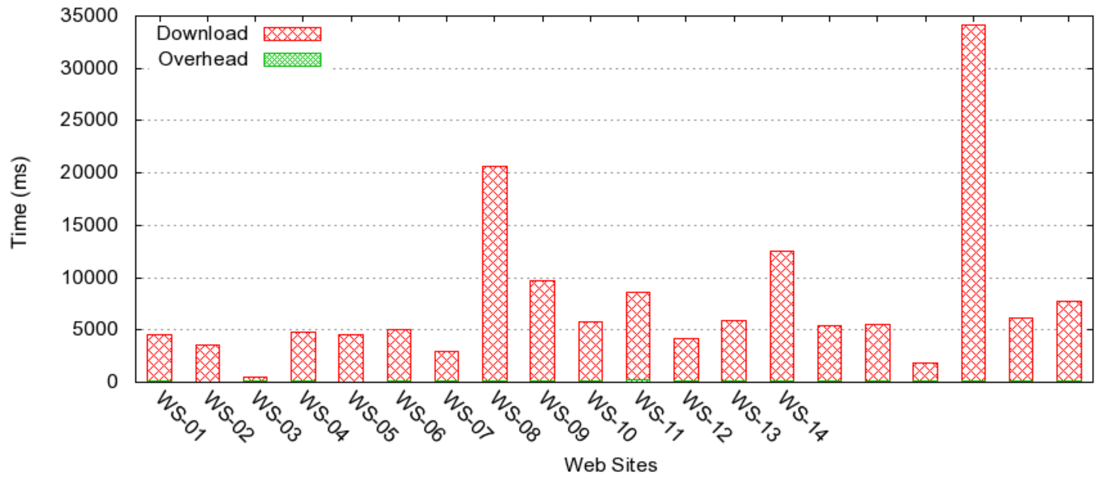
Number of Users	***LAN Delays (%)***	***Internet Delays (%)***
2	2.0 ± 1.0	3.2 ± 1.3
3	2.1 ± 1.7	2.0 ± 0.5
4	–	2.8 ± 0.7
5	4.4 ± 2.2	2.6 ± 1.1
7	3.7 ± 3.5	3.2 ± 1.7
9	2.4 ± 1.4	2.2 ± 1.0
10	2.7 ± 1.9	–
12	–	2.8 ± 1.0
15	–	3.5 ± 1.1
16	8.7 ± 8.2	–
19	3.5 ± 2.7	–
20	–	2.1 ± 0.8
22	4.8 ± 3.7	–

compared to $6,170msec$ (local network). This fact can be explained by the heterogeneity of the users' connection to the Internet and their geographical distance to the co-browsed web servers. Still in the case with 9 users, comparing only the notification protocol delay obtained in both scenarios ($Internet = 419 \pm 183msec$ and $LAN = 149 \pm 87msec$), the difference between them (270 ± 270) is essentially due to the users' network delay ($270 \pm 270 \approx 230 \pm 150$) to reach our server. In other words, the delay added by OCEAN is affected mainly by the physical distance to the OCEAN application server and the network conditions, a common effect in distributed systems.

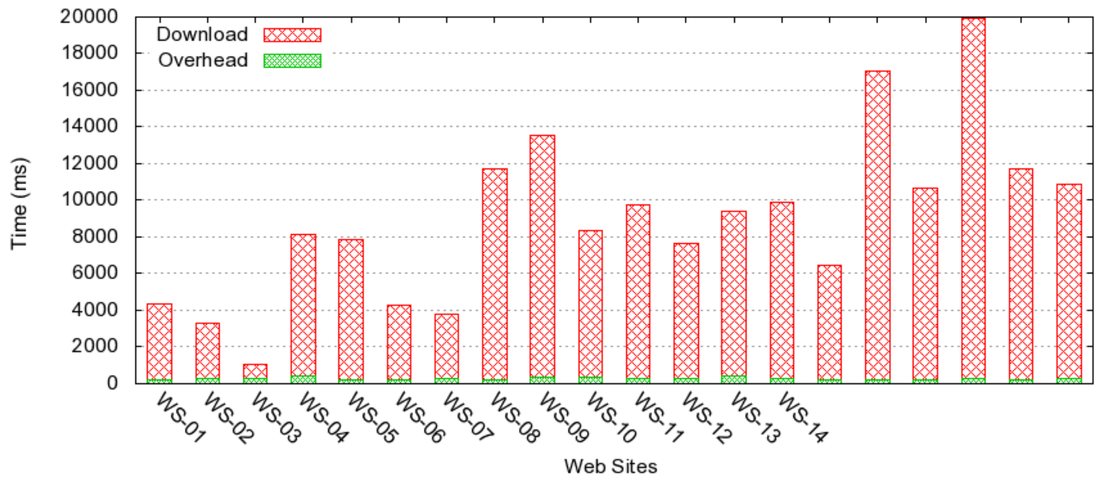
The traditional navigation (single mode) paradigm is naturally affected by the conditions of a heterogeneous environment. Due to the design of OCEAN's architecture, users perform web content download as if they were in single mode, thus being identically affected. The only difference between OCEAN's navigation and single mode navigation is due to the protocol delay. However, as the results discussed above have shown, considering the total asynchrony period, this delay is short enough keeping its value around 3.8% for the LAN scenario, and 2.7% for the Internet. Thus, OCEAN keeps its lightweight performance in both scenarios.

6.1.2 Web Pages Effect

Someone could ask: *If I co-browse to a huge web page, then notification protocol delay would proportionally increase, becoming significant?* In order to investigate such delay from this new perspective, similar experiments were conducted, focusing now on how this delay varies as the web page size increases. These new experiments consist on performing measurements from local and distributed scenarios, but considering more web sites (20) and less users (from 1 to 6). The obtained results are summarized on Figure 6.3.



(a) LAN - *from: 03/19/2009 05:30 PM , to: 03/19/2009 09:00 PM*



(b) Internet - *from: 03/29/2009 00:10 AM , to: 03/29/2009 03:30 AM*

Figure 6.3: *Proportional Delay - Web Pages Parameter*: a report of medium *notification protocol delay* and *longest download times* observed in experiments, that have focused on the effect of web pages different sizes over such measures.

Similarly to the previous experiments, these column charts show the short delay imposed by the notification protocol, against the total asynchrony period. Moreover, these web sites (*WS-**) are ordered by their sizes, as detailed in Table 6.2. It is important to note that besides ordered by web page size, the obtained results do not present a regular growing behavior regarding the download time. Such behavior is due to the fact that download time is also affected by other factors than the web page size, where some relevant are the time for preparing dynamic web pages[7], and the limitations at the web server, for instance, due to server capacity or some network congestion.

Table 6.2: Experiments Data - Web Pages Parameter: details about each experimented web site.

Index	***URL***	***Size (KB)***	***Proportional Delays (%)***	
			LAN	*Internet*
WS-01	http://ppgi.inf.ufes.br	44	6.4 ± 2.2	5.0 ± 1.6
WS-02	http://gwt-widget.sourceforge.net/	44	1.9 ± 4.3	8.3 ± 3.3
WS-03	http://www.pageshare.com	98	16.6 ± 1.0	25.8 ± 3.0
WS-04	http://code.google.com	99	1.9 ± 1.0	4.1 ± 1.8
WS-05	http://ieeexplore.ieee.org/search /wrapper.jsp?arnumber=4575153	112	1.3 ± 1.7	2.1 ± 1.6
WS-06	http://www.cookiecentral.com/	149	1.5 ± 2.8	5.4 ± 2.0
WS-07	http://sahi.co.in/w/	180	2.8 ± 2.8	7.2 ± 2.1
WS-08	http://www.uv.mx	183	1.0 ± 1.0	1.8 ± 1.3
WS-09	http://www.ewi.utwente.nl/en/	190	1.0 ± 1.3	2.3 ± 0.9
WS-10	http://www.technologyreview.it	215	1.3 ± 0.9	3.6 ± 2.9
WS-11	http://www.mercadolivre.com.br	218	1.9 ± 0.4	2.6 ± 0.8
WS-12	http://www.eclipse.org	232	2.3 ± 0.6	3.2 ± 2.6
WS-13	http://www.kiobo.com/	296	1.7 ± 11.2	3.8 ± 16.7
WS-14	http://www.laas.fr/laas	299	1.0 ± 1.3	2.8 ± 5.7
WS-15	http://www.w3schools.com /browsers/browsers_stats.asp	317	1.9 ± 1.0	3.5 ± 2.3
WS-16	http://www.informatik.uni-erlangen.de/	374	1.7 ± 1.6	1.2 ± 4.8
WS-17	http://www2.ufscar.br/home/index.php	441	6.0 ± 0.3	2.3 ± 3.8
WS-18	http://slashdot.org	445	0.5 ± 1.0	1.2 ± 2.6
WS-19	http://www.yahoo.com.br	566	4.1 ± 0.9	2.1 ± 2.3
WS-20	http://www.g1.com.br	684	1.8 ± 0.4	2.2 ± 1.5

[7]Web pages built dynamically in accordance with requests or users' sessions parameters. However, it demands a process time in the web server before delivering its content to the client.

These charts suggest us a smoothness of the delay measured values, regardless the size of co-browsed web-site. Such smoothness can be better observed in Figure 6.4, where only the delay measured values are depicted. Indeed, this relative stable behavior of the notification protocol delay measures was expected because the OCEAN's Notification Protocol does not handle the web pages' content, but only their URLs composes the exchanged messages. Consequently, assuming a low probability of network congestion, then the protocol delay is only affected by the length of such URL strings which has usually few bytes. Therefore, the actions of co-browsing for huge or tiny web pages do not affect the designed notification process, thus this design characteristic holds the respective notification protocol delays.

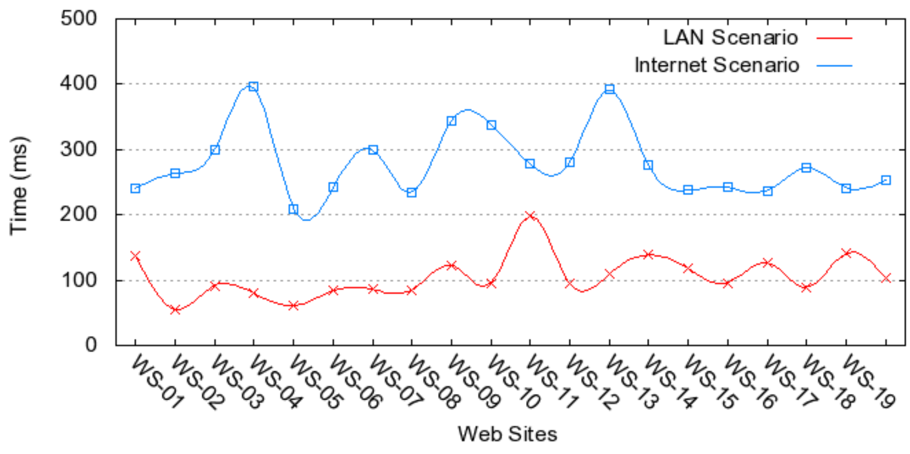


Figure 6.4: *Delay Comparison - Web Pages Parameter*: a comparison of the observed delay measures from both evaluation distribution scenarios (laboratory and Internet).

In particular, the average gap between the delay observed in both scenarios is $153.46 \pm 62.40 msec$. As aforementioned, these differences should be mainly due to network delays, including propagation, transmissions and queues delays. The distributed users have presented and average network delay of 53.65 ± 4.26 while in the local network we have observed 0.29 ± 0.08. Thus, we can see that the observed gap is basically the difference between the scenarios' specific propagation delays. As a consequence, this network delay is the mainly component of the network distribution which affects our system. However, even being affected by users distance to our application server, the experiments showed that the Notification Protocol still provides an acceptable performance, since it imposes a protocol delay which tends to be imperceptible to users.

6.2 Application Server Scalability

As mentioned in the beginning of this chapter, the OCEAN Application Server intermediates every user interaction in the system. However due to its centralized design, this server could become a bottleneck considering OCEAN performance. This way, this section focuses on evaluating this server, in order to determine if this architectural component is scalable, which means, how much workload this server is able to support without damaging the responsiveness.

Apart from previous section, the adopted experiment-centered evaluation approach is not enough for supporting this scalability evaluation task now. That is because our experiments were limited by a very relevant constraint, the number of available client machines.

In an effort to overcome this problem, this section relies on a more elaborated evaluation methodology. Here, data collected in experiments with few real clients feed an analytical model, as the ones treated in (69). Such modeling is adopted in an effort to predict the server performance with a large number of users, without using a large number of real clients.

Menasce & Almeida(69) have proposed a simple methodology for evaluating web services' capacity. A subset of this methodology is used in the scope of this work, and can be summarized on three main phases. The first is related to knowing the environment, which consists of discovering what kind of hardware (clients and servers), software (operating systems, middleware and applications), network connectivity and communication protocols are present in the environment. Next two phases are related to constructing models for the evaluation task. The *workload model* captures the resource demand and the characteristic workload intensity for each component of a global workload, inside a relevant time interval. While the *performance model* is used for forecasting or predicting if the target system will offer performance measures that would satisfy established service level agreements.

6.2.1 Knowing the Environment

Before describing the environment used in this study, it is important to make some remarks. In this evaluation of OCEAN's performance, we are interested in discovering limitations of our proposal. With this in mind, we have turned our attention to worst possible configurations that involve server performance. Due to that, our analysis is restricted by two main parameters. Firstly, we consider all users participating in only one co-browsing thread. We believe that this constraint forces clients to make more requests at the same time, for instance, download acknowledgement messages. At last, we evaluate the server scalability especially in a LAN scenario. This decision favors simultaneous requests arrivals at the server.

Regarding systems' environment, we have adopted the same components used on LAN scenario of notification protocol delay experiments. Such scenario consists of:

Hardware: The OCEAN Application Server is held is a non-dedicated platform, which comprises a Intel Xeon QuadCore $X3220$ $2.40GHz$ $64bits$ and $4GB$ of RAM (random access memory). Likewise, the client application runs on non-dedicated machines, generally composed by Intel Celeron, Pentium 3 and Pentium 4 processors and containing from $128MB$ to $1GB$ of RAM.

Software: Even that OCEAN does not impose operating system preference requirements, both server and clients are evaluated on Linux 2.6.* platforms. In particular, the application server is deployed on an Apache Tomcat 6.0.18, running on a Sun Java 1.5.0_12 platform. Our service also relies on a database management system for storing co-browsing resources. For performing such storage task we have chosen a PostgreSQL 8.3, using Hibernate4GWT framework as an abstraction layer.

Network Connectivity: The local network is organized in a way that the server machine acts as OCEAN application server as well as the LAN router, connecting this LAN to the university's network ($5Mbps$) and therefore to the Internet. Inside the LAN, all client machines are connected to a switch $10Mbps$, which is connected to the network router.

Communication Protocol: The great majority of network communications in OCEAN is supported by the same basic protocol stack: Notification Protocol, GWT-RPC (60)(63), HTTP, TCP and others overlaid in the specific networks.

6.2.2 Workload Model

The workload of a system can be defined as the set of all input information received by this system during an specific time period. However, it is a daunting task to handle real workloads with great number of elements. So, on working with practical problems, usually becomes necessary to reduce and summarize such workload information. In other words, it is needed to create a workload model that captures the most relevant characteristics of the real workload (69).

Considering the investigated environment and our main objective, that is to evaluate the OCEAN Application Server, we need to identify the basic components of the workload and all the related information that are relevant to our objective. For instance, understanding how the service in focus handles contribution messages and, obtaining quantitative measures about clients' requests to this service.

The first step towards the workload model is the workload characterization. Such characterization is the process of accurately describing the global workload of the system, in terms of its components. In this matter, the OCEAN's workload is characterized regarding two basic perspectives: (*i*) client-server interactions, referred as request classes; and (*ii*) basic components compounding the server.

Considering that the most relevant client-server interactions in OCEAN are supported by the Notification Protocol, the request classes for this workload model are the two phases of such protocol, *send* and *receive*. These two interaction modes clearly have distinct server demands (see section 4.2.3), due to that, are a good choice to distinguish the workload. Moreover, another point that worth to consider is participant roles. These roles, *presenter* and *attendee*, defined by the Privilege Management Policy (section 4.2.2) affect all communications. For example, depending on the publisher role, a published contribution can have greater server resources demands, for delivering to the right subscribers. Roles also distinguish, especially, what kind of contributions an interaction could contain. Summarizing, considering protocol's phases and privilege roles, we have defined four request classes: *presenter-send*, *presenter-receive*, *attendee-send* and *attendee-receive*. These request classes were distinguished from the global workload, in an effort to understand their specific needs, and how well they have been satisfied.

The specified request classes distinguish types of inputs received by the server. However, the server itself also must be distinguished by means of its basic components or resources. This approach is useful, for instance, for determining which of these components is a bottleneck, that damages the server scalability. This way, we could have a indication of where the service must be improved in order to provide a better scalability. In this workload model, the two server basic components are hardware-based. They are the *cpu*, representing the processing demands, and the *i/o* component, representing input and output demands, for instance, database accesses. Besides these two basic resources, due to implementation issues we have included a third one, the *sleep* component.

As aforementioned, in the publish-subscribe paradigm, the distribution service delivers new messages to subscribers through server-push mechanisms (58). However, such strategy is difficult to implement as a pure server-push without the server being able to reach all the clients. In this matter, a server-push is usually emulated along polling techniques (60). By the same reason, OCEAN contribution receive phase is implemented using a polling technique. For this prototype, such feature was implemented as a client infinite loop procedure, where on each loop step this client makes a request (protocol receive phase) to the server looking for new published contributions. On the server side, if there is not any novelty, such requests are put on a *sleep* condition, waiting until a new contribution arrives or until this request reaches a timeout. Concerning this implementation issue, we decide distinguish this time spent in the sleep condition, in order to identify the true demands of receive requests.

Therefore, the OCEAN workload model constructed for this study is based on three basic components (*cpu*, *i/o* and *sleep*), and is partitioned on four request classes (*presenter-send*, *presenter-receive*, *attendee-send* and *attendee-receive*). Thus, to complete this model it is necessary to determine some characterization parameters. The workload intensity refers to the number of users participating simultaneously in the same co-browsing thread. While, the last parameters to consider are the components' specific demands, in other words, the average resource demands for each request class (69).

These parameters are defined and validated using collected data during the same LAN experiments conducted for determining the protocol delay. During such experiments, we have collected the time each request spent on each component. Through these observations, we could determine the OCEAN service demands, presented in Table 6.3. In particular, details about how these data were collected, can be found in Section A.2.

Table 6.3: Service Demands Matrix: this table represents the OCEAN's workload model used in this performance study, containing the service demands of components by each request class.

Basic Components	***Request Classes***			
	presenter-send	*presenter-receive*	*attendee-send*	*attendee-receive*
CPU	3.92 *msec*	1.07 *msec*	0.01 *msec*	0.95 *msec*
I/O (database)	5.69 *msec*	0.00 *msec**	2.26 *msec*	0.00 *msec**
Polling Sleep	0.00 *msec**	6,632.90 *msec*	0.00 *msec**	10,440.57 *msec*

* Requests of the related class do not use the referred server resource.

The service demand matrix summarizes the workload model, containing a reduced set of representative elements. Such model now represents the whole real workload of OCEAN, and can be used for predicting its performance.

6.2.3 Performance Model

Performance forecasting is the process of estimating the performance measures of a computer system for a given set of parameters. Examples of these parameters are network protocols, load balancing disciplines, resources limitations, workload intensity and service demands (69). Moreover, this forecasting task needs to use models, that are essentially distinguished on two main modeling techniques: *simulation* and *analytical* modeling. Both are abstractions of the reality, representing the systems in study by their most relevant elements. Basically, the simulation modeling technique comprises implementing a software that simulates the behavior of the target systems' main components. Conversely, analytical modeling concerns a set of mathematical equations built to capture such behavior.

Considering that analytical models can be a cost-effective alternative to provide relatively quick answers to "what-if" questions giving more insights for the system being studied (4), we have chosen to use an analytical modeling technique based on queuing networks for representing the competition for resources on the OCEAN application server. This modeling activity starts by finding an analytical model that well represents the system, and if necessary modifying it.

The model illustrated in Figure 6.5 is designed to approximately fit OCEAN architecture and workload characteristics using a *closed queuing network* containing the main resources of the server(69), more precisely the OCEAN application server. In particular, a queuing network is considered *closed*, where all the request classes can be considered *closed classes*, in which there are always a known number of instances of each request class existing in the system. In our case, these numbers are functions of the number of participants joined in the co-browsing thread.

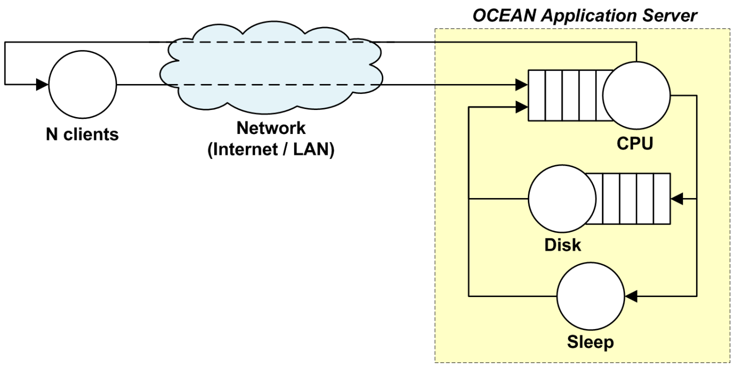


Figure 6.5: Performance Model: a closed queuing network representing the most relevant concepts for this analytical modeling study.

The dependency relations between the number of requests in the system (closed queuing network) and the number of thread group members are expressed in table 6.4. We mainly consider browsing related contributions (*web browse* and *download ack*) in this evaluation study, as a consequence these dependency values are derived from the privilege grants of such contributions. In other words, since one and only one of the thread group members (n) is a presenter while the others ($n-1$) are attendees, and considering the fact that each user maintain two independent requests in the system (*send* and *receive*), therefore the presenter maintains one *presenter-send* and one *presenter-receive* requests in the system, whereas the body of all attendees ($n-1$ users) maintain, in the same way, $n-1$ *attendee-send* and $n-1$ *attendee-receive* requests in the system.

Table 6.4: Dependency Between Numbers of Requests and Participants: request class specific relation considering that all session participants (n) joined the same co-browsing thread.

Request Class	***presenter-send***	***presenter-receive***	***attendee-send***	***attendee-receive***
Requests in the System	1	1	$n-1$	$n-1$

Having chosen an analytical model, it must be solved in order to provide some performance forecasting results. The technique adopted for solving this closed queue network is MVA (Mean Value Analysis)(70). This MVA technique is essentially an iterative algorithm that combines the model equations and the workload service demands, in order to reach values of residence time, throughput and queuing lengths. At each iteration, the MVA algorithm decreases the error on the values, until becomes lower then a specified tolerance value. The MVA equations for the OCEAN component level model are listed in figure 6.6. Additionally, in order to avoid the complexity of an deterministic solution of a multi-class closed queuing networks, we have adopted an MVA approximation proposed by Schwitzer (71). More details about the MVA technique and the derivation of those equations can be found on (69).

Therefore, the listed equations can be used to solve the analytical model, for instance: (*i*) *service demands* (table 6.3); (*ii*) number of simultaneous requests in the system (table 6.4); and (*iii*) the *error tolerance value*, used for stopping algorithm iterations[8]. Thus, for completing the model solution, it is just necessary to reevaluate the MVA for each number of participants in the co-browsing thread, since the other two algorithm parameters are constants or functions of the number of users.

6.2.4 Analytical Model Results

Figure 6.7 depicts a throughput perspective of the analytical model results. In this case, we have an indication for predicting how many simultaneous contributions the server supports as the number of participants increases. A typical throughput curve presents a rapid growing of the throughput value, until the system reaches the saturation point. This is the case of *attendee-send* class (Figure 6.7(b)) that reaches the saturation point handling $434.3tps$, around $50users$ participating in the co-browsing thread.

[8]In this work, we have set this error tolerance to 10^{-5}, for solving the model.

Residence time equation for request class r at queue i:

$$R'_{i,r}(\overrightarrow{N}) = \begin{cases} D_{i,r} & \text{delay resource;} \\ D_{i,r}[1 + n_i(\overrightarrow{N} - \overrightarrow{1_r})] & \text{queuing resource.} \end{cases} \tag{6.2}$$

Throughput equation for request class r:

$$X_{0,r}(\overrightarrow{N}) = \frac{N_r}{\sum_{i=1}^{K} R'_{i,r}(\overrightarrow{N})} \tag{6.3}$$

Queue length equation for request class r at queue i:

$$n_{i,r}(\overrightarrow{N}) = X_{0,r}(\overrightarrow{N}) \times R'_{i,r}(\overrightarrow{N}) \tag{6.4}$$

Queue length equation for queue i:

$$n_i(\overrightarrow{N}) = \sum_{i=1}^{R} n_{i,r}(\overrightarrow{N}) \tag{6.5}$$

Schweitzer's approximation:

$$n_i(\overrightarrow{N} - \overrightarrow{1_r}) = \frac{N_r - 1}{N_r} n_{i,r}(\overrightarrow{N}) + \sum_{t=1 \& t \neq r}^{R} n_{i,t(\overrightarrow{N})} \tag{6.6}$$

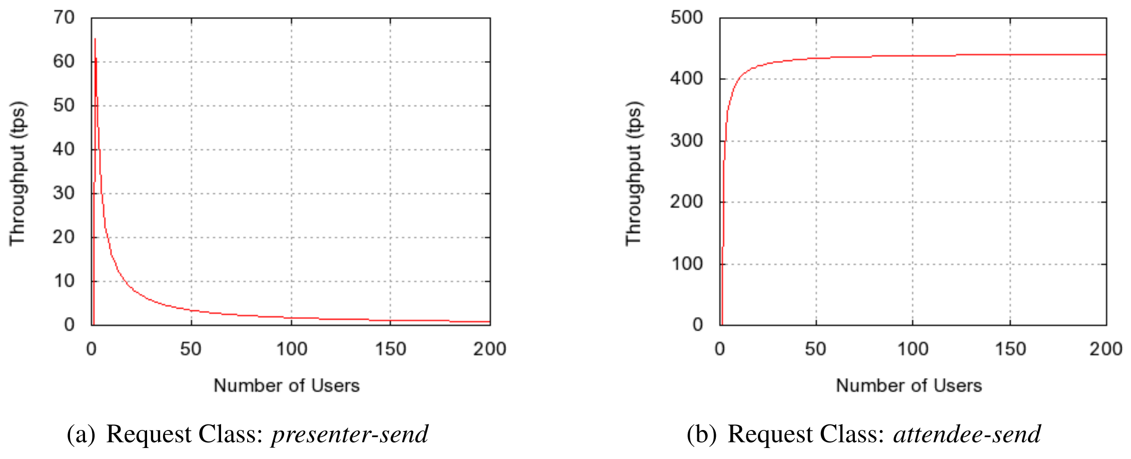
Iteration error:

$$\varepsilon = max_{i,r} \left| \frac{n^e_{i,r}(\overrightarrow{N}) - n_{i,r}(\overrightarrow{N})}{n^e_{i,r}(\overrightarrow{N})} \right| \tag{6.7}$$

Figure 6.6: MVA Equations: equations that guide the MVA algorithm on solving the component level model (69).

However, the *presenter-send* class (Figure 6.7(a)) presents such an uncommon throughput curve. In this case, the saturation point is already with 1 user. It is due to growth dependency between the number of requests and the number of users (Table 6.4), since there is always just one request of this class in the system, not mattering the number of participants in the co-browsing thread. Considering now $100users$, the application server still could handle $1.7tps$. As the *web browse* is the only *informative contribution* composing this request class, such throughput value is acceptable, since the relative response time for supporting $100users$ is around $588msec$.

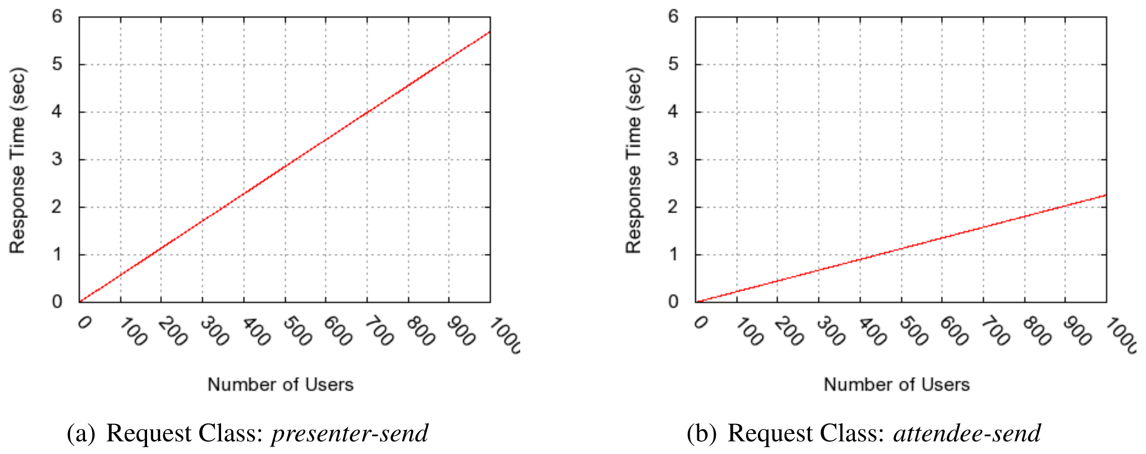
Another analysis perspective of the obtained results is concerning the expected response times. It is worth to remember that this metric is the main focus of the chapter. As expected for any system, as simultaneous requests in the system increases, the average response time tends to increase too. This behavior is shown by figure 6.8, which presents a forecast of response



(a) Request Class: *presenter-send* (b) Request Class: *attendee-send*

Figure 6.7: Analytical Model Results - Throughput

times.

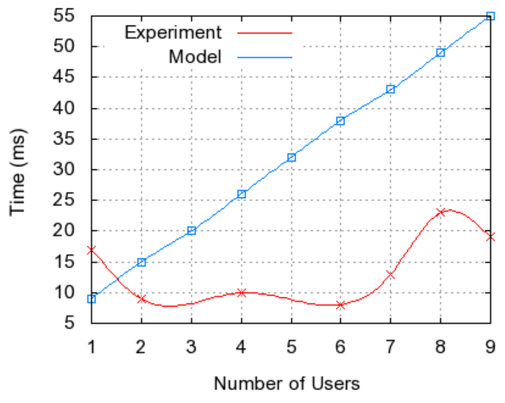


(a) Request Class: *presenter-send* (b) Request Class: *attendee-send*

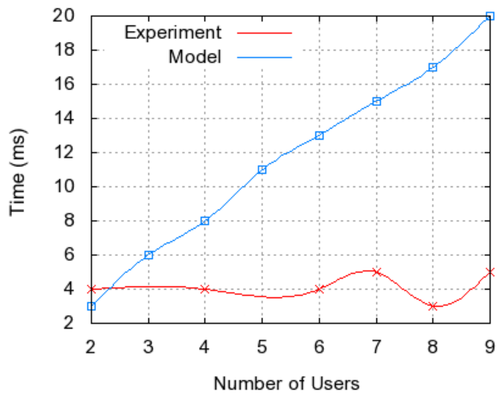
Figure 6.8: Analytical Model Results - Response Time

Assuming that the infra-structure for supporting a co-browse service is composed by only one server, we consider that the system provides an acceptable level of scalability. As it can be seen, even in such a impracticable case, having 500 users participating in the same co-browsing session, the application server still is able to handle requests in an acceptable time, where the *presenter-send* requests would be handled in 2.8 seconds, while the *attendee-send* in 1.1 second.

As a final remark, comparing the results obtained from the performance model and the practical experiments, we have observed that the performance model has not only overestimated the responsiveness of OCEAN, but also it had an inaccurate behavior (Figure 6.9). A possible explanation for that is because of adopted instrumentation method (see Appendix A), used for conducting the experiments. Recall that the parameters (*service demands*) used as entries to the analytical model were collected using an instrumentation method executed at the application level. At the application level, some measurement points are inserted at the server to take the instant a request arrives, goes through the server resources (CPU, disk and sleep) and leaves. However, we recognize that there can be some imprecision on estimating service demand. Even though, considering this inaccuracy the obtained results still are valuable and the performance model tends to be an upper bound for response time.



(a) Request Class: *presenter-send*



(b) Request Class: *attendee-send*

Figure 6.9: Validating the Performance Model: these graphs compare the results about server response time, obtained from experiments and from solved performance model.

6.3 Conclusions

This chapter showed that OCEAN's design provides acceptable responsiveness to users. The delay imposed by the co-browsing mechanism has a low cost allowing users to experience a high level of synchronization in co-browsing sessions. Following, the application server evaluation suggests that the co-browsing system scales suited to users which is a very important characteristic for turning OCEAN's prototype into a usable release candidate.

The adopted methodologies during such performance evaluation task involved two of the three basic disciplines of performance analysis: *experimentation* an *analytical modeling*. The first included a great effort on making repetitive experiments with the variation of some parameters (number of users, size of web sites and type of distributed users: dense or sparse). Coordinating such experiments was constrained by an inconvenience, which was the number of available computer for performing tests. The obtained results of notification protocol delay has demonstrated the high performance archieved by our prototype.

The second study has had a poor accuracy of the chosen model, and also the unexpected effect of the sleep component which has led to less relevant results than we expected. However, these results were still useful for providing us a better indication of the server scalability, rather than intuitive insights. Therefore, such study can be very helpful for guiding further attempts of evaluation OCEAN.

For future work, there are others characteristics that should be evaluated and if necessary improved in OCEAN. Still considering performance, there are relevant issues to evaluate, for example, the effect of simultaneous co-browsing threads, or even simultaneous sessions in the system. Moreover, it will be interesting to extend the assumption that the application server does not fail, then we could investigate for instance: (*i*) *system dependability* (4); (*ii*) *usability*, regarding the quality of system usage perceived by individuals and groups(15); and (*iii*) the impact of such usage on organizations.

7 General Conclusions and Outlooks

This work proposed an integrated solution for collaborative web browsing, taking into account the intrinsic characteristics of this collaboration paradigm, its main issues usually faced on building processes of groupwares dedicated to it. As a result, we have proposed OCEAN, a comprehensive groupware specification and implementation, developed for addressing the main requirements involved in a common co-browsing scenario.

Therefore, this chapter is dedicated to conclude this work providing a review of the obtained results, highlighting its main contributions, as well as presenting relevant topics to be considered in future work.

7.1 Revisiting our Goals

The main goal of this thesis has been defined (Section 1.2) as: "*Propose an environment that fits users needs for collaboratively browsing the web with arbitrary purposes*".

We expect to have fulfilled this objective by proposing OCEAN. Its features were specified in an effort to cover all the collaboration aspects (cooperation, coordination and communication) in order to offer solutions for the different activities related to a general co-browsing teamwork. In addition to that, OCEAN's design and implementation have considered performance (responsiveness and scalability) and usability (user interface) requirements. As a consequence, the obtained results prove OCEAN to fulfil "*users needs*", given that it provides adequate tools (proposed features) with an efficient usage (design and implementation) for "*collaboratively browsing the web with arbitrary purposes*".

Furthermore, the subsequent specific goals that have been addressed:

- *Goal 1*: "Provide *flexibility*, a property that we pursuit in all aspects of this work. It refers to the design of collaboration environments which are adaptable to users usage needs, as well as to the environment conditions".

OCEAN has taken into account *flexibility* in different aspects. The first is on the coordination mechanism, which allows users to freely contribute to co-browsing sessions, browsing web pages independently, or synchronized in groups (*thread groups*). This mechanism allows users to use co-browsing facilities for many purposes. For instance, in a rigid virtual lecture or on collaboratively searching information on the web. Secondly, the notification protocol is completely managed by the *privileges policy*, and because of that, it can be specialized by simple modifications at the privilege grants.

Another relevant characteristic is the *extensibility* of OCEAN, which increases the flexibility of the proposal itself. This characteristic is particularly present at the definition of an *informative contribution* (Section 3.2). In other words, all communications in the system is made through notification of informative contributions. So, if someone could add a new feature (with similar QoS requirements) on OCEAN, he or she just needs to specialize this concept, extending the privileges table.

- *Goal 2*: "Keep our work in accordance with well-founded collaboration theories, in an effort to produce broader and consistent results while avoiding mistakes during the development process".

  We have based this work on the 3C model (17) and the Collaboration Ontology (19). These complementary theories mainly helped us on understanding "what" was the problem that we were dealing with. Especially, the Collaboration Ontology offered an expressive knowledge about the collaboration domain. Thus, OCEAN could be defined inline with general collaboration characteristics.

- *Goal 3*: "Design an architecture capable maintaining a high synchronization perception for users, while they collaboratively browse the web".

  The proposed architecture (Chapter 4) includes the *application server*, performing the minimal actions for maintaining synchronized sessions; the *clients*, taking advantage of standard browsers; and the *notification protocol* allows to maintain the last two elements loosely-coupled. Such architecture was defined aiming at providing acceptable response time perceived by users and system scalability, supporting a lightweight system performance. Indeed, the performed evaluations (Chapter 6) have shown that this architectural design provides a satisfactory response time, since the imposed delays are usually negligible. In addition, the performance analysis of the prototype indicates the acceptable level of scalability with respect to the application server, which is the most critical element of the whole architecture. All these results and the awareness features included in the proposal, suggest to a "*high synchronization perception for users*".

- *Goal 4*: "Adapt common features of traditional web browsing to this collaborative context. This way, users could feel more comfortable when using such environment".

  The implemented prototype (Chapter 5) was modeled organizing the proposed co-browsing features, in a similar way of a standard browser. Amongst them there are: (*i*) *address bar*, providing a way for choosing web sites to co-browse, besides accessing co-browsing thread URL histories; (*ii*) *browser tabs*, organizing different browsing contexts (co-browsing threads); and (*iii*) *co-browsing history*, containing all the artifacts produced in the session.

  Another important characteristic of OCEAN is the annotations feature. Although not related to standard browsers, this feature waves the co-browsing sessions more efficient, providing a simple mechanism for users to get involved in the teamwork.

7.2 Contributions

The main contributions of the work presented in the thesis are summarized in the following:

1. *Co-browsing Paradigm Review* - Most of the work in this area (e.g.: (11)(8)(12)(41)) define co-browsing through simples variations of the same extremely general statement: "jointly browsing the web". However, they propose solutions focusing on specific interpretations or subsets of this definition.

   In this matter, the first contribution of this work is on reviewing many related works that deals with co-browsing, and as a result composing a more precise specification of this paradigm and its main variations (Section 2.2). The co-browsing definition proposed in this work considers co-browsing an act of recommending web contents, and many variations of this paradigm can be summarized by five main characteristics: recommendation method, interaction synchronism, users location requirements, co-browsing purpose and coupling level.

2. *The Proposed Features* - The features proposed in OCEAN covered the three collaboration aspects, turning a very expressive system, but without loosing the focus on the collaborative web browsing paradigm.

   Regarding coordination, the specified co-browsing threads permit users co-browsing on the desired "manageable freedom" state. In other words, users can freely contribute without damaging the session. So, this mechanism provides a balanced coordination for OCEAN. Additionally, its primitives and also the way they were implemented (*browser tabs* and *privilege tokens*) allow efficient and simple management of co-browsing sessions.

At the cooperation perspective, the proposed co-browsing history represents the shared production space which maintains users together, even when working from different physical places. A similar feature can be encountered in (6), however, this solution is sensitive to session growth, generating confusion. In OCEAN, such problem is minimized by contextualizing history artifacts according to nested scopes (*session*, *thread* and *checkpoint*).

Finally, considering communication, the annotation feature appears as a good choice for a co-browsing specialized feature, since it is tightly related to the most important handled artifacts (the web pages). This characteristic is especially important for the efficiency of communication actions, avoiding for instance misinterpretations. But, is spite of such high specialization degree, the annotations feature is general enough for being useful on almost any co-browsing scenario.

3. *Towards a Methodology for Groupwares Development* - The Collaboration Ontology can be seen as an specialization of the 3C model, providing a more detailed and accurate view about the collaboration domain. Thus, we propose the usage of this theory during the development process of groupware. Such framework could be used also in the initial phases of this process, despite others proposals which only provides reusable architectural components. Amongst the main benefits of such domain ontology modeling, there are: avoiding mistaken modeling decisions, rapid prototyping and easier interoperability with another collaborative systems.

4. *Comparison of Design Approaches* - The architectural issues discussed in Section 4.1 provide a detailed comparison of the most commonly adopted solutions, the content and browser manipulation techniques. In particular, it considers technical characteristics that were rarely discussed in other references.

5. *A Well-defined and Extensible Protocol* - The notification protocol has been an advantage of OCEAN, providing an unique and relatively simple way of maintaining distributed users synchronized. The publish/subscribe paradigm fitted suitably with the communication requirements (*i.e.*: responsiveness and loosely-coupling) and with the nature of supported contributions characteristics (*e.g.*: rare and delivery guaranteed). More important, the

privilege policy turns this protocol extremely extensible and configurable, thus providing flexibility for co-browsing sessions.

6. *Insights on Performance Evaluation* - Although not providing the most satisfactory results, the employed evaluation process is itself a contribution, especially in the CSCW research area, where these kinds of evaluation is not frequently performed. This work has provided different methodologies for evaluating performance characteristics, and showing how to proceed in such evaluation tasks (see Appendix A). This knowledge could be valuable for improving another groupware proposals.

7.3 Open Questions and Future Work

Several directions can be explored for future research to extend the contributions presented in this work, either overcoming imposed constraints during OCEAN's development process, or extending its proposal and goals themselves.

Considering the characterization of the proposal (Chapter 3), the most relevant limitation was with respect to the supported features. It could be interesting to support more features in the future. This could be made by implementing features from the scratch, extending OCEAN's model, or even integrating external applications, taking advantage of the collaboration domain ontology.

Concerning systems' design (Chapter 4), new features and contributions can be easily embedded in the notification protocol. However, we are aware that this protocol is not a general solution for every type of communication in a groupware. For instance, and audio call certainly would lose quality if it were transmitted using the notification protocol. In this matter, it becomes necessary to investigate the QoS (Quality of Service) requirements of each proposed feature, and when a new one is to be inserted, it must be investigated too. The first step of such possible evolution should lie on offering different service levels at the notification protocol itself, in the same way as in (62), distinguishing communication channels by priority transmissions. Such options could preserve the benefits of notification protocol and the privilege grants.

On the sequence, considered that OCEAN's implementation (Chapter 5) still is a prototype, there are a lot of work to get it complete. For example, we have not worked on porting the browser extension to other standard browsers besides Mozilla Firefox. Another improvement in OCEAN is with respect to turning the browser tabs usage to more more intuitive and practical. Additionally, in a short-term is it would would be valuable to allow a better reuse of co-browsing histories, especially after the session over. A feature like that could promote for instance: asynchronous co-browsing sessions, distribution of virtual lecture notes, meeting reports, etc. Therefore, such reuse of an already stored information (co-browsing history) would significantly expand the scope of OCEAN usages and capabilities.

At last, the evaluation process (Chapter6) has faced inconvenient problems, which leads to some limitations at the obtained results. The most affected was the scalability study. Even so, the obtained results, mainly at the notification protocol delay study have shown the performance quality of OCEAN. Thus, for mid-term research and development, it would be valuable to refine the evaluation instruments and methods, in an effort to obtain more accurate and relevant results.

7.4 Final Considerations

In face of all obtained results we consider OCEAN a successful attempt on producing a comprehensive and objective collaborative web browsing environment. More important, this process has produced scientific and technological contributions to the CSCW research field.

APPENDIX A – Experimentation Practices

Chapter 6 presented the evaluation of OCEAN, which was totally based on repeatedly conducting experiments in order to quantify the performance proportionated by the proposed design. However, this chapter does not detail the experimentation process. In this appendix, we intend to provide a more detailed view about how the experiments have been conducted, especially showing the OCEAN prototype instrumentations, developed for capturing measures relevant for the performance evaluation task.

In the following, Section A.1 details how the application client was instrumented for capturing the necessary information for determining the delay imposed by the notification protocol. In the sequence, Section A.2 presents the instrumentations in the application server, made for collecting input to the analytical model. Finally, Section A.3 describes a simple automatic test agent implemented for helping on the experiments.

A.1 Client Instrumentation

The OCEAN application client has been instrumented aiming at quantifying the notification protocol delay. Remaining to the equation for calculating such a value (Equation 6.1), the required parameters are the *asynchrony period* and the *highest download time*. Considering a thread group, only the presenter participant, more precisely his/her client application, is the only entity that has the exact notion of how long is an asynchrony period. This is because the presenter is responsible for browsing, in other words, he/she starts the asynchrony periods. Also, since this period ends at the arrival of the last download ack, thus all these client instrumentations are focused on providing information to this presenter participant.

At first, every thread group member (their clients) measures the time spent downloading the web page. Such time starts when the client application sets the content panel's *iframe* source attribute with the published URL, and finishes when this *iframe* triggers a *load* event. In doing so, this measured time includes the time spent downloading, for instance, the HTML file and other embedded files as images or scripts.

In the sequence, the attendees' client applications append this information as a message of the *download ack*. This way, at the end of the co-browsing cycle[1] the presenter's client gathers the measured download time of all thread group members, since it knows its own download time. In addition to that, the presenter's client also stores the moment that have start publishing the *web browse* contribution, and also the moment that it receives each *download ack*. Having all this information, the presenter's client is able to calculate the notification protocol delay value. This process is illustrated by Figure 6.1.

In order to maintain an organized repository of experiments, the presenter sends all the calculated delays and the related parameters for an additional servlet[2] in the OCEAN application server, at the end of the session.

A.2 Server Instrumentation

Following the performance evaluation approach, the OCEAN application server has been instrumented aiming at providing input for the adopted analytical model, and thus quantifying the server scalability. In other words, at every request to this server, we get measures about the demands of such requests on the server components (CPU, Disk, Sleep).

All the HTTP requests regarding the notification protocol are directed to the same servlet published at the OCEAN application server, which runs in the Apache Tomcat. This servlet implements the Notification Service, a component of the OCEAN architecture presented in Section 4.1.2. Thus, for determining the demands of our service, we have instrumented this specific servlet, observing every request it handles.

The server instrumentation is composed by a two level observation method. At the first level, we have developed the *Measurements Filter*, a servlet filter[3] responsible for observing the requests as they were atomic operations, obtaining data related to their response times and content lengths. These observed data feed a log denominated the *Request Level Report*. More important, this servlet filter intercepts the requests at the right moment they arrive at the Apache Tomcat, before they being decoded and handled by the Notification Service, and thus link with the respective response just before it leaves the server towards the client.

[1]Co-browsing cycle is the sequence of interactions that happen in an asynchrony period, depicted by figure 4.3

[2]A Java class that dynamically handle requests in a HTTP server. In our case this server is an Apache Tomcat (tomcat.apache.org).

[3]The Java Servlet specification version 2.3 introduces a new component type, called a *filter*. A *filter* dynamically intercepts requests and responses to transform or use the information contained in the requests or responses. Filters typically do not themselves create responses, but instead provide universal functions that can be "attached" to any type of servlet or JSP page (72).

At the second level, we focus on measuring the specific demands of server components, due to that, we have augmented the OCEAN Notification Service source code (servlet) with measurement points. In doing so, at each handled request the notification service feeds another log (*Application Level Report*) with the observed demands for each server component. Moreover, at this level, we are able to know what is the related request class, since it is possible to access the content of the request decoded by the GWT-RPC layer [4].

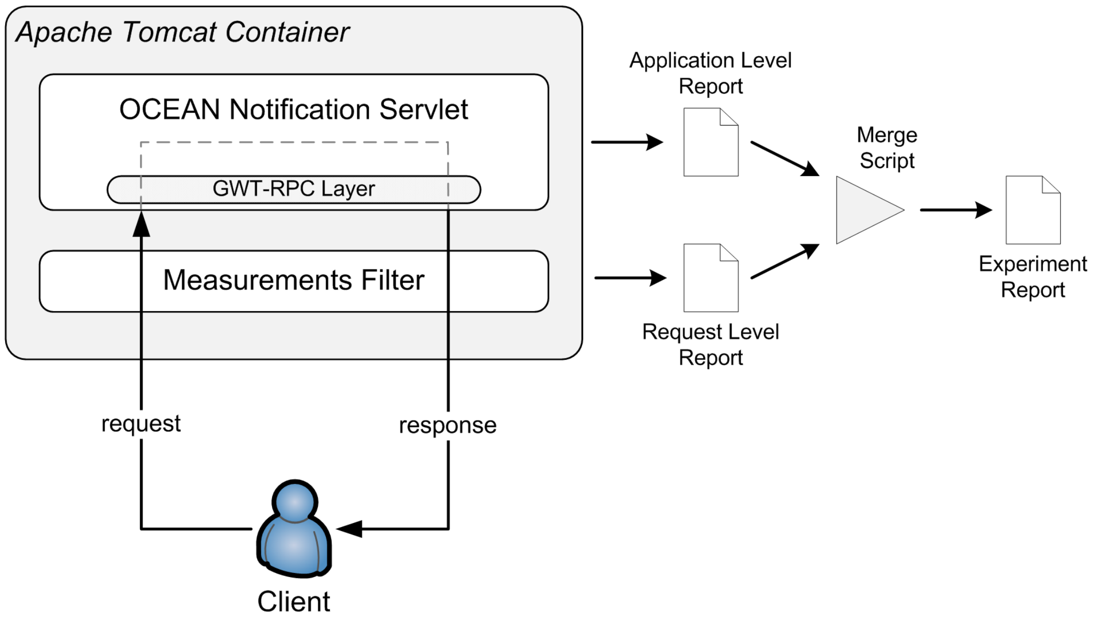


Figure A.1: Server Measurements Structure

Figure A.1 depicts this two-level observation method. Note that, the *Measurements Filter* has been created aiming at having more accurate measures of response times, that would consider the whole time spent in the server, including transparent processes for managing GWT-RPC connections. This way, using servlet filters our measures could turn our results more realistic. However, the servlet filter itself do not achieve all measurements goals, since its necessary to classify measurements according to request classes and performance components, information only available inside our notification service implementation, after request decoding by the GWT-RPC layer. In this matter we make that two parallel observation efforts, and after the end of the co-browsing session we merge the two produced reports into the *Experiment Report*, which contains a detailed evaluation of each request combining information of both previous reports. This *Experiment Report* provides therefore, the necessary input for the performance model and thus the scalability study presented in Section 6.2.

[4]Since we have implemented OCEAN prototype based on the GWT framework (63), the RPC communication are encoded in accordance with the GWT-RPC specification. This way, this framework offers a transparent layer in the client and server implementations for automatically managing the RPC connections and transmitted data.

A.3 Remote Test Agents

On performing experiments with real applications it is a hard task to mobilize too many people for helping on all necessary tests. Therefore is common to implement automatic agents for acting as simple clients performing the most basic tasks necessary for the experiment. As a consequence, it becomes easy to perform high scale tests, not depending of too many volunteers.

In our experiments we have followed the same approach, implementing an automatic agent for performing two basic tasks, joining a default thread group and act as a regular attendee, which includes receiving *web browse* contributions, downloading referred web pages and thus publishing *download acks*. This way it is just necessary one real user conducting the experiments, acting as the presenter of the co-browsing thread.

Our first intention was to develop a java stand-alone application that would reuse the client-side libraries provided by the GWT framework, adapting the implementation of the notification protocol send and receive functions (see Section 4.2.3). However, at the used version of GWT framework (version 1.4) we do not encountered a feasible way for reusing such libraries out of a browser JavaScript engine. However, the cost of re-implementing the entire GWT-RPC specification only for this test purpose would not worthwhile. Due to that, instead of re-implementing too many processes for building automatic agent as a simple java stand-alone application, we have made it dependent of a browser. In other words, we have augmented the OCEAN application client with characteristics of the desired automatic agent, to be used whether we need to perform experiments.

When accessing the prototype login web page (Figure 5.1), it is possible to inform HTTP parameters that will make this client in particular to behave as an automatic agent, only performing the aforementioned basic tasks. Illustrating this agent initialization procedure, let's suppose that the address of OCEAN application were `http://ocean/`. In this case, for turning a client into an automatic agent, it would just be necessary to append parameters to the address, like: `http://ocean/?agentName=A1&agentEmail=A1@lab.ufes&sessionID=e815512d5`. Doing that, the application client automatically joins the first available thread group in the session `e815512d5`, becoming a regular attendee in it. Therefore, for performing tests it is just necessary to have one real user which creates the thread group and participate as a presenter, and also triggers all other attendees as automatic agents.

Such agent triggering process is made by an implemented script[5] which establish SSH (Secure Shell) connections with all target machines. Through this connections, this script

[5]This agent management script is implemented in Python, combining Linux shell scripts.

sends a complete Mozilla Firefox installation already containing the OCEAN browser extension installed. After decompressing this browser, the script run it remotely accessing the OCEAN login web page with the respective agent parameters. Illustrating this process, the shell script command executed for starting an arbitrary agent *A1* is depicted in the following.

```
expect -c "set timeout -1;spawn ssh <AGENT_HOST> -l <SSH_USERNAME> \"
  cd /tmp/ocean_experiments/;
  tar -zxf firefox_ocean_pack.tar.gz;
  export MOZ_CRASHREPORTER_DISABLE ;
  firefox/firefox -no-remote --display=:0 -CreateProfile Ocean_Agent;
  firefox/firefox -no-remote --display=:0 -P Ocean_Agent
    http://ocean/?agentName=A1&agentEmail=A1@lab.ufes&sessionID=e815512d5;
\";match_max 100000;expect *assword:*;send -- <SSH_PASSWORD>\r;interact;"
```

In addition to that, the experiments were usually composed by repeated tests accessing the same set of web sites. This way, the OCEAN application client was also augmented with and automatic browse list. In other words, the presenter has access to a special button capable of automatically publishing web browse contributions, following a pre-defined list. And more important, waiting until the last *download ack* of a co-browsing cycle arrives, before iterating to the next web site of such list.

A last modification made for automating test was the finalization of the test sessions. In accordance with OCEAN specification and design, the users have the right to *log out* of the session whenever they want. However the automatic agents are not supposed to know when would be the better moment for logging out. In this matter, the notification protocol has been extended in terms of adding a new contribution type, the *force agent logout contribution*. This contribution can be published by a presenter at any time (using another special button), and are propagated to all attendees joined in the same thread group. On receiving this contribution, the attendees' clients automatically log out the co-browsing session, without require any approval of the user. This contribution is quite useful for properly finalizing the co-browsing session at the OCEAN application server. Moreover, it would be useful in the future on defining features like *kick user* or *ban user*, well-known coordination facilities at group environments.

References

1 SANTOS, R. O.; OLIVEIRA, F. F.; ANTUNES, J. C. P.; MARTINELLO, M.; GUIZZARDI, R. S. S.; GOMES, R. L. Licob: Lightweight collaborative browsing. In: *Workshop Web2Touch*. Milan, Italy: [s.n.], 2009. [submitted].

2 OLIVEIRA, F. F.; ANTUNES, J. C. P.; GUIZZARDI, R. S. S. Towards a collaboration ontology. In: GUIZZARDI, G.; FARIAS, C. (Ed.). *Proceedings of the 2nd Workshop on Ontologies and Metamodels in Software and Data Engineering (WOMSDE'07)*. [S.l.: s.n.], 2007. João Pessoa, Brazil.

3 SANTOS, R. O.; MARTINELLO, M.; MARCONDES, C.; FABRIS, F. Joinus: Management of mobile social networks for pervasive collaboration. In: SBC. *Proceedings of the 5th Simpósio Brasileiro de Sistemas Colaborativos (SBSC'08)*. [S.l.]: IEEE Computer Society, 2008. p. 224–234. ISBN 978-0-7695-3500-5. Vila Velha, Brazil. [doi: 10.1109/SBSC.2008.14].

4 MARTINELLO, M. *Availability Modeling and Evaluation of Web-based Services - A Pragmatic Approach*. PhD Thesis — Institut National Polytechnique de Toulouse, Toulouse, France, 2005. Available at: <www2.laas.fr/laas/1-4266-Publications.php>.

5 AMERSHI, S.; MORRIS, M. R. Cosearch: a system for co-located collaborative web search. In: *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI'08)*. New York, NY, USA: ACM, 2008. p. 1647–1656. ISBN 9781605580111. [doi: 10.1145/1357054.1357311].

6 ANEIROS, M.; ESTIVILL-CASTRO, V. Usability of real-time unconstrained www-co-browsing for educational settings. In: *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*. Washington, DC, USA: IEEE Computer Society, 2005. p. 105–111. ISBN 0-7695-2415-X. [doi: 10.1109/WI.2005.154].

7 BROOKS, C.; HANSEN, C.; GREER, J. Social awareness in the ihelp courses learning content management system. In: *Workshop on the Social Navigation and Community based Adaptation Technologies*. [S.l.: s.n.], 2006.

8 GEROSA, L.; GIORDANI, A.; RONCHETTI, M.; SOLLER, A.; STEVENS, R. Symmetric synchronous collaborative navigation. In: *IADIS International Conference WWW/Internet*. [S.l.: s.n.], 2004. p. 748–754.

9 DIEBERGER, A.; DOURISH, P.; HööK, K.; RESNICK, P.; WEXELBLAT, A. Social navigation: techniques for building more usable systems. *Interactions*, ACM, New York, NY, USA, v. 7, n. 6, p. 36–45, 2000. ISSN 1072-5520. [doi: 10.1145/352580.352587].

10 Sosign Interactif. *Clavardon: A co-browsing tool for e-commerce*. 2008. Project website: <www.clavardon.com>. Release September 2008.

11 HOYOS-RIVERA, G. J.; GOMES, R. L.; WILLRICH, R. C.; COURTIAT, J. P. Colab a new paradigm and tool for browsing collaboratively the web. *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, v. 36, n. 6, p. 1074–1085, 2006. [doi: 10.1109/TSMCA.2006.883173].

12 ESENTHER, A. Instant co-browsing: Lightweight real-time collaborative web browsing. In: *The Eleventh International World Wide Web Conference*. Honolulu, Hawaii: [s.n.], 2002.

13 SANTOS, R. O.; SANA, D. M.; OLIVEIRA, F. F. *Blocool: Share your blog reading, discover new blogs*. March 2009. Project website: <www.blocool.com>. Release: 0.4.

14 MAINTAINERS. *Browzmi: Web Together*. 2008. Project website: <www.browzmi.com>. Release September 2008.

15 NEALE, D. C.; CARROLL, J. M.; ROSSON, M. B. Evaluating computer-supported cooperative work: models and frameworks. In: *Proceedings of the 2004 ACM conference on Computer-Supported Cooperative Work (CSCW'04)*. New York, NY, USA: ACM, 2004. p. 112–121. ISBN 1-58113-810-5. Chicago, Illinois, USA. [doi: 10.1145/1031607.1031626].

16 PINELLE, D.; GUTWIN, C. A review of groupware evaluations. In: *Proceedings of the 9th IEEE International Workshops on Enabling Technologies (WETICE'00)*. Washington, DC, USA: IEEE Computer Society, 2000. p. 86–91. ISBN 0-7695-0798-0.

17 ELLIS, C. A.; GIBBS, S. J.; REIN, G. Groupware: some issues and experiences. *Communications ACM*, ACM, New York, NY, USA, v. 34, n. 1, p. 39–58, 1991. ISSN 0001-0782. [doi: 10.1145/99977.99987].

18 FUKS, H.; RAPOSO, A.; GEROSA, M. A.; LUCENA, C. J. P. Applying the 3c-model to groupware engineering. *International Journal of Cooperative Information Systems (IJCIS)*, v. 14, n. 2-3, p. 299–328, Jun-Sep 2005.

19 OLIVEIRA, F. F. de. *Uma Teoria Ontólogica de Colaboração e suas Aplicações no Domínio de Colaboração*. Master Thesis — Universidade Federal do Espírito Santo, Vitória, Brazil, 2009. [to appear].

20 BORGHOFF, U. M.; SCHLICHTER, J. H. *Computer-Supported Cooperative Work: Introduction to Distributed Applications*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2000. ISBN 3540669841.

21 FARIAS, C. R. G. de. *Architectural Design of Groupware Systems: a Component-Based Approach*. PhD Thesis — University of Twente, The Netherlands, 2002. Available at: <http://purl.org/utwente/37999>.

22 MALCHER, M. A. da G.; ENDLER, M. A context-aware collaborative presentation system for handhelds. In: SBC. *Proceedings of the 5th Simpósio Brasileiro de Sistemas Colaborativos (SBSC'08)*. [S.l.]: IEEE Computer Society, 2008. p. 1–11. Vila Velha, Brazil. [doi: 10.1109/SBSC.2008.13].

23 CISCO. *WebEx - Web Conferencing and Collaboration Solutions*. 1992. Project website: webex.com.br. Release March 2009.

24 ADOBE. *Acrobat Connect Pro*. 2009. Project website: www.adobe.com/products/acrobatconnectpro. Release April 2009.

25 FUKS, H.; RAPOSO, A.; GEROSA, M. A.; PIMENTEL, M.; FILIPPO, D.; LUCENA, C. J. P. Inter- e intra-relações entre comunicação, coordenação e cooperação. In: *Proceeding of th 4th Simpósio Brasileiro de Sistemas Colaborativos*. Rio de Janeiro, RJ, Brazil: SBC, 2007. p. 83–96. ISBN 978-85-7669-126-6.

26 GUTWIN, C.; GREENBERG, S. Effects of awareness support on groupware usability. In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI'98)*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1998. p. 511–518. ISBN 0-201-30987-4. [doi: 10.1145/274644.274713].

27 CABRI, G.; LEONARDI, L.; ZAMBONELLI, F. A proxy-based framework to support synchronous cooperation on the web. *Software Practice Experience*, John Wiley & Sons, Inc., New York, NY, USA, v. 29, n. 14, p. 1241–1263, 1999. ISSN 0038-0644. [doi: 10.1002/(SICI)1097-024X(19991210)29:14<1241::AID-SPE277>3.0.CO;2-V].

28 Kiobo. *Kiobo Social Browsing*. 2008. Project website: <www.kiobo.com>. Release September 2008.

29 CHANG, M. L. *CoBrowse*. 2006. Project website: <cobrowse.mozdev.org>. Release August 2008.

30 HOYOS-RIVERA, G. de J. *CoLab - Conception et Mise Ňuvre d'un Outil pour la Navigation Coopérative sur le Web*. PhD Thesis — Université Paul Sabatier, Toulouse, France, 2005. Available at: <www2.laas.fr/laas/1-4266-Publications.php>.

31 PageShare Technologies Inc. *PageShare*. 2008. Project website: <www.pageshare.com>. Release September 2008.

32 MAINTAINERS. *BlogRollr: What blog posts have you been reading?* 2009. Project website: blogrollr.com. Release March 2009.

33 MAINTAINERS. *Blog Scrobbler*. 2009. Project website: code.google.com/p/blogscrobbler. Release March 2009.

34 WANG, W.; HAAKE, J. M. Flexible coordination with cooperative hypermedia. In: *Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space—structure in hypermedia systems (HYPERTEXT'98)*. New York, NY, USA: ACM, 1998. p. 245–255.

35 SCHMIDT, K.; SIMONE, C. Coordination mechanisms: towards a conceptual foundation of cscw systems design. *Computer Supported Cooperative Work*, Kluwer Academic Publishers, Norwell, MA, USA, v. 5, n. 2-3, p. 155–200, 1996. ISSN 0925-9724. [doi: 10.1007/BF00133655].

36 MALY, K.; ZUBAIR, M.; LI, L. *CoBrowser: Surfing the Web Using A Standard Browser*. Norfolk, VA, USA, 2000.

37 NGUYEN, A. V.; REKIK, Y.; GILLET, D. A framework for sustaining the continuity of interaction in Web-based learning environment for engineering education. In: *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA 2005)*. Montreal, Canada: [s.n.], 2005.

38 SCHMIDT, K.; BANNON, L. Taking cscw seriously: Supporting articulation work. *Computer Supported Cooperative Work*, v. 1, p. 7–40, 1992.

39 KHEZAMI, N.; OTMANE, S.; MALLERR, M. An approach to modelling collaborative teleoperation. In: *Proceedings of the 12th International Conference on Advanced Robotics (ICAR'05)*. [S.l.: s.n.], 2005. p. 788–795. [doi: 10.1109/ICAR.2005.1507498].

40 RAPOSO, A.; FUKS, H. Defining task interdependencies and coordination mechanisms for collaborative systems. *Frontiers in Artificial Intelligence and Applications*, IOS Press, Amsterdam, NL, v. 74, p. 88–173, 2002.

41 LIMA, C. V.; WILLRICH, R.; GOMES, R. L.; HOYOS-RIVERA, G. de J.; COURTIAT, J.-P. A co-browsing system with conference support. *Scientia - Interdiciplinary Studies in Computer Science*, v. 18, n. 2, p. 79–96, July/December 2007.

42 Google Inc. *OpenSocial*. 2008. Available at: <code.google.com/apis/opensocial>.

43 GOMES, R. L. *LEICA - Un environnement faiblement couplé pour l'intégration d'applications collaboratives*. PhD Thesis — Université Paul Sabatier, Toulouse, France, 2006. Available at: <http://tel.archives-ouvertes.fr/tel-00088729>.

44 MARSHALL, C. C.; BRUSH, A. J. B. Exploring the relationship between personal and public annotations. In: *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries (JCDL'04)*. New York, NY, USA: ACM, 2004. p. 349–357. ISBN 1-58113-832-6. [doi: 10.1145/996350.996432].

45 PROVENSI, L. L.; COSTA, F. M.; SACRAMENTO, V. Tinta digital em aplicações multimídia para ambientes móveis. In: SBC. *Proceedings of the XIV Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia'08)*. Vila Velha, ES, Brazil: IEEE Computer Society, 2008. p. 49–52.

46 CHONG, N. S. T.; SAKAUCHI, M. Creating and sharing web notes via a standard browser. *SIGCUE Outlook*, ACM, New York, NY, USA, v. 27, n. 3, p. 4–15, 2001. ISSN 0163-5735. [doi: 10.1145/504546.504547].

47 LAURILLAU, Y.; NIGAY, L. Clover architecture for groupware. In: *Proceedings of the 2002 ACM conference on Computer supported cooperative work (CSCW'02)*. New York, NY, USA: ACM, 2002. p. 236–245. ISBN 1-58113-560-2. [doi: 10.1145/587078.587112].

48 RICARDO, J. M. *Um Framework Para Construção Cooperativa de Ambientes do tipo CSCW/CSCL*. PhD Thesis — Universidade Federal do Espírito Santo, Vitória, Brazil, 2004.

49 NARDI, J. C. *Apoio de Gerência de Conhecimento à Engenharia de Requisitos em um Ambiente de Desenvolvimento de Software*. Master Thesis — Universidade Federal do Espírito Santo, Vitória, Brazil, 2006.

50 GUARINO, N. Understanding, building and using ontologies. *Int. J. of Human-Computer Studies*, v. 46, n. 2-3, p. 293–310, 1997. [doi: 10.1006/ijhc.1996.0091].

51 SMITH, B.; WELTY, C. Ontology: Towards a new synthesis. In: SMITH, B.; WELTY, C. (Ed.). *Proc. of the 2nd International Conf. on Formal ontology in information systems*. New York: ACM Press, 2001. p. 3–9.

52 GAAEVIC, D.; DJURIC, D.; DEVEDZIC, V.; SELIC, B. *Model Driven Architecture and Ontology Development*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN 3540321802.

53 WANG, X.; CHAN, C. W.; HAMILTON, H. J. Design of knowledge-based systems with the ontology-domain-system approach. In: *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*. New York, NY, USA: ACM, 2002. p. 233–236. ISBN 1-58113-556-4.

54 GUARINO, N. Formal ontology and information systems. In: GUARINO, N. (Ed.). *Proc. of the 1st Formal Ontology and Information Systems*. Trento, Italy: IOS Press, 1998. (1st), p. 3–15.

55 MAINTAINERS. *CoBrowser.net*. 2008. Project website: <www.cobrowser.net>. Release July 2008.

56 MALY, K.; ZUBAIR, M.; LI, L. Cobrowser: Surfing the web using a standard browser. In: *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*. Chesapeake, VA: AACE, 2001. p. 1220–1225.

57 MAINTAINERS. *Cookie Central*. Available at: <www.cookiecentral.com/>. Last Access: January 2009.

58 EUGSTER, P. T.; FELBER, P. A.; GUERRAOUI, R.; KERMARREC, A.-M. The many faces of publish/subscribe. *ACM Computing Surveys*, ACM, New York, NY, USA, v. 35, n. 2, p. 114–131, June 2003. ISSN 0360-0300. [doi: 10.1145/857076.857078].

59 DOMMEL, H.-P.; GARCIA-LUNA-ACEVES, J. J. Floor control for multimedia conferencing and collaboration. *Multimedia Systems*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 5, n. 1, p. 23–38, 1997. ISSN 0942-4962. [doi: 10.1007/s005300050040].

60 HANSON, R.; TACY, A. *GWT in Action: Easy Ajax with the Google Web Toolkit*. Greenwich, CT, USA: Manning Publications Co., 2007. ISBN 1933988231.

61 MARCONDES, C.; MARTINELLO, M.; SCHWARTZ, R. S.; SANTOS, R. O.; SANADIDI, M.; GERLA, M. Pathcrawler: Automatic harvesting web infra-structure. In: *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'08)*. [S.l.]: IEEE Computer Society, 2008. p. 339–346. ISBN 978-1-4244-2065-0. ISSN 1542-1201. Salvador, Brazil. [doi: 10.1109/NOMS.2008.4575153].

62 DYCK, J.; GUTWIN, C.; GRAHAM, T. C. N.; PINELLE, D. Beyond the lan: techniques from network games for improving groupware performance. In: *Proceedings of the 2007 international ACM conference on Supporting group work (GROUP'07)*. New York, NY, USA: ACM, 2007. p. 291–300. ISBN 978-1-59593-845-9. [doi: 10.1145/1316624.1316669].

63 Google Inc. *GWT: Google Web Toolkit*. 2008. Available at: <code.google.com/webtoolkit/>. Release: 1.4.

64 HANSON, R. *GWT Widget Library*. 2006. Available at: <gwt-widget.sourceforge.net>.

65 W3Schools.com. *Browser Statistics*. 2009. Available at: <www.w3schools.com/browsers/browsers_stats.asp>. Last Access: March 2009.

66 AT Internet Institute. *Browsers Barometer*. 2009. Available at: <www.atinternet-institute.com/en-us/browsers-barometer/index-1-2-3-0.html>. Last Access: April 2009.

67 MAINTAINERS. *Sahi -Web Automation and Test Tool*. 2005. Available at: <sahi.co.in/w/>. Last Access: March 2009.

68 MAINTAINERS. *iMacros*. 2000. <www.iopus.com/imacros>. Last Access: March 2009.

69 MENASCE, D. A.; ALMEIDA, V. A. F. *Capacity Planning for Web Services: metrics, models, and methods*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN 0130659037.

70 REISER, M.; LAVENBERG, S. S. Mean-value analysis of closed multichain queuing networks. *J. ACM*, ACM, New York, NY, USA, v. 27, n. 2, p. 313–322, 1980. ISSN 0004-5411. [doi: 10.1145/322186.322195].

71 SCHWEITZER, P. Approximate analysis of multiclass closed networks of queues. In: *Proceddings of the Internation Conference on Stochastic Control and Optimization*. [S.l.: s.n.], 1979.

72 Sun Microsystems Inc. *The Essentials of Filters*. 2009. Available at: java.sun.com/products/servlet/Filters.html. Last Access: April 2009.