



CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS  
DEL INSTITUTO POLITÉCNICO NACIONAL

**Departamento de Ingeniería Eléctrica**  
**Sección de Computación**

**Entorno Deductivo y Auto-Adaptativo de Autoría  
Cooperativa en la Web**

Tesis que presenta  
**Muhammad Aslam**  
para obtener el Grado de  
**Doctor en Ciencias**  
en la Especialidad de  
**Ingeniería Eléctrica**

Directora de la Tesis:

**Dra. Ana María Martínez Enríquez**

México, D.F.

Octubre 05 del 2005



CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS  
DEL INSTITUTO POLITÉCNICO NACIONAL

**Electrical Engineering Department**  
**Computer Science Section**

**A Deductive and Self-Adaptive Web Cooperative  
Authoring Environment**

Presented by

**Muhammad Aslam**

as the fulfillment of the requirement for the degree of

**Doctor of Science**

Specialization in

**Electrical Engineering**

Option:

**Computer Science**

Advisor:

**Dra. Ana María Martínez Enríquez**

Mexico City, Mexico.

October 05, 2005

# Contents

<b>Resumen</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>Acknowledgments</b>	<b>xi</b>
<b>Abbreviations</b>	<b>xiii</b>
<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xvi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Cooperative Authoring with AllianceWeb . . . . .	6
1.3 PIÑAS: a Middleware for Web Cooperative Authoring . . . . .	8
1.4 The Distributed Event Management Service . . . . .	9
1.5 Group Awareness Management . . . . .	11
1.6 Design and Implementation Principles of the Inference Engine . . . . .	13
1.7 Main Contribution . . . . .	14
1.8 Thesis Overview . . . . .	15
<b>2 Related Domains to Workgroup Computing</b>	<b>17</b>
2.1 Computer Supported Cooperative Work . . . . .	17
2.2 Groupware . . . . .	18
2.3 Awareness . . . . .	19
2.3.1 Group Awareness . . . . .	20
2.3.2 Action, Task, and Activity Awareness . . . . .	21
2.3.3 Workspace Awareness . . . . .	21
2.3.4 Peripheral Awareness . . . . .	22
2.3.5 Contextual Awareness . . . . .	22
2.3.6 Conversational Awareness . . . . .	23
2.3.7 People, Document and Collection Awareness . . . . .	23
2.3.8 Perspective Awareness . . . . .	23

2.3.9	Synchronous/Asynchronous Awareness . . . . .	24
2.3.10	Presence/Social Awareness . . . . .	24
2.4	Privacy, Intrusiveness, Security, and Awareness . . . . .	25
2.5	Communication and Awareness Support . . . . .	26
2.6	Coordination and Awareness Support . . . . .	28
2.7	Conclusions . . . . .	29
<b>3</b>	<b>A Group Awareness Inference Engine to Enhance Cooperative Work</b>	<b>31</b>
3.1	The Structure of a Cooperative Writing Application . . . . .	32
3.2	The Architecture of AllianceWeb . . . . .	33
3.3	Principles of the Group Awareness Inference Engine . . . . .	37
3.4	Goals of the Group Awareness Inference Engine . . . . .	38
3.5	Enhancing Group Awareness . . . . .	39
3.5.1	Work Focus and Perception . . . . .	41
3.5.2	Work Proximity . . . . .	46
3.5.3	Opening-Closing Document and Starting-Leaving Session . . . . .	48
3.5.4	Notifying Inactiveness within a Cooperative Session . . . . .	50
3.5.5	Privacy and Intrusion . . . . .	52
3.5.6	Classifying a Shared Document . . . . .	52
3.6	Conclusions . . . . .	56
<b>4</b>	<b>Transformation of MathML Formulas</b>	<b>59</b>
4.1	Introduction . . . . .	59
4.2	MathML formulas . . . . .	60
4.2.1	Producing a MathML Formula . . . . .	61
4.2.2	Handling MathML Events . . . . .	61
4.2.3	Well-formed MathML Formulas . . . . .	62
4.3	Transformation of a MathML Formula . . . . .	64
4.3.1	Single Component Pattern . . . . .	65
4.3.2	Twofold Component Pattern . . . . .	68
4.3.3	Threefold Component Pattern . . . . .	71
4.4	Prefix and Postfix Notation . . . . .	76
4.5	The Scalable Vector Graphics, SVG Schema . . . . .	76
4.6	Conclusion . . . . .	78
<b>5</b>	<b>Evaluation of Cooperative Production</b>	<b>81</b>
5.1	Introduction . . . . .	81
5.2	Evaluation of MathML Expressions . . . . .	82
5.2.1	The Complexity of a Formula . . . . .	82
5.2.2	Undo/redo Action . . . . .	85
5.2.3	Elapsed_Time . . . . .	87
5.2.4	Evolution of the Production . . . . .	89
5.3	Presence Awareness and Assistance . . . . .	92

*CONTENTS*

v

5.3.1	Deduction of Experts . . . . .	94
5.4	Some Related Work . . . . .	94
5.5	Conclusions . . . . .	97
<b>6</b>	<b>Conclusions</b>	<b>99</b>
<b>A</b>	<b>Syntax of the GAIE Language</b>	<b>103</b>
<b>B</b>	<b>The GAIE Rule Base</b>	<b>105</b>
<b>C</b>	<b>GAIE Functions</b>	<b>121</b>
<b>D</b>	<b>GAIE Actions</b>	<b>123</b>
<b>E</b>	<b>Glossary</b>	<b>127</b>



# Resumen

Esta investigación se enfoca en el diseño y el desarrollo de un sistema de conciencia de grupo (group awareness) para soportar la interacción/producción cooperativa de manera eficiente. La falta de conciencia de grupo en la autoría cooperativa es la principal motivación de este trabajo de investigación. Validamos nuestro enfoque en la aplicación de coautoría basada en la Web: AllianceWeb. Las funcionalidades de conciencia de grupo fueron desarrolladas siguiendo los principios de la Inteligencia Artificial, particularmente los sistemas basados en conocimiento. Nuestra contribución está enmarcada en tres líneas principales: 1) una comunicación basada en contexto para permitir a los coautores precisar el foco de discusión ("*work focus*"), 2) la coherencia de la producción cooperativa para permitir a los coautores estar conciente de las secciones interrelacionadas del documento ("*work proximity*"), 3) la evaluación de la producción cooperativa para deducir/cualificar que usuarios son posibles expertos, con la finalidad de determinar coautores quienes necesitan asistencia y coautores quienes pueden eficientemente ayudar a los demás.

Otro aspecto principal de este trabajo es la focalización en proveer funciones de reestructuración de alto nivel para facilitar al usuario transformar/reutilizar fórmulas MathML.





# Abstract

This research focuses on the design and the development of a group awareness system to support cooperative productions/interactions in an efficient way. The lack of group awareness in cooperative authoring is the principle motivation of this work. We validate our approach into a Web-based cooperative authoring application: AllianceWeb. The awareness functionalities are developed following Artificial Intelligence principles, particularly those of the knowledge based systems. Our contributions are provided in three main lines: 1) the context-based communication to allow coauthors to precise the "*work focus*" under discussion, 2) the consistency of the cooperative production to allow users to be aware of the interrelated sections of the document ("*work proximity*"), 3) the cooperative production evaluation to deduce/qualify which users are possible experts, in order to determine coauthors who need assistance and coauthors who may efficiently help others.

The other main aspect of this work is to focalize on providing high level restructuring functions to facilitate a user to transform/reuse MathML formulas.



# Acknowledgments

The advice, assistance and friendship of a number of people has been invaluable during my stay at CINVESTAV-IPN.

First and foremost, my most heartfelt thanks are due to Dra. Ana María Martínez Enríquez for making me welcome to this institution, providing continued guidance, and support throughout. Her opinion, experience and insight were of great assistance for me. It is all due to her polite behavior that I am at this stage.

I am in debit with Dr. Dominique Decouchant for countless invaluable discussions, ideas, practical support and for being a constant motivating force that helped me reviewing this research to conclusion.

I wish to acknowledge the very helpful comments and suggestions from my five excellent reviewers, who dedicated much time and effort to finalize this manuscript: Dr. Ernesto Suaste Gómez, Dr. Jesús Favela, Dr. José Oscar Olmedo Aguirre, Dra. Xiaou Li, and Dr. Hugo Sánchez Salguero.

This work would not have been possible without the support of my parents, who throughout my academic progression, were always a source of encouragement that made me strive to keep improving and embrace new challenges. I am deeply indebted to my life partner and my kids. I particularly wish to thank them for never complaining when they needed my company.

I thank to all my friends with whom I lived for their valuable support during the whole period of this research; Joselito Medina Medina, Eduardo Martínez Paniagua in particular, and other companions of Computer Section in general.

Finally, I am grateful to the Ministry of Foreign Affairs, Mexico (SRE), Ministry of Education, Islamabad, Pakistan, and Embassy of Pakistan, Mexico for awarding me the Cultural Exchange Scholarship (UAC-III, No. 604743, Exp:811.(72:00)/137) that supported me financially for four years. In this respect, I am particularly thankful to CINVESTAV-IPN (Scholarship to terminate the doctorate), the Chairman of the department of Electrical Engineering, and the Chairman of Computer Section (Project No. 45306Y-Conacyt) for facilitating me final scholarships that allowed me to conclude this research without any economical strain and stress.

Zacatenco, Mexico City, Mexico, October 05, 2005

Muhammad Aslam

E-mail: [muhammad@computacion.cs.cinvestav.mx](mailto:muhammad@computacion.cs.cinvestav.mx)

Dedicated to  
my parents, my life partner, and my kids

# Abbreviations

BSCW	Basic Support for Cooperative Work
CSCW	Computer Supported Cooperative Work
DEMS	Distributed Event Manager Service
DES	Digital Encryption Standard
EMISARI	Emergency Management Information System and Reference Index
EPS	Encapsulate Postscript
EF	Expertise Finder
ER	Expertise Recommender
GAIE	Group Awareness Inference Engine
HTML	Hyper Text Markup Language
JPG	Joint PhotoGraphic expert
IM&P	Internet instant Messaging and Presence
MathML	Mathematical Markup Language
MEDD	Multistrategy Error Detection and Discovery
MSN	MicroSoft Network
PDA	Personal Digital Agent
PDF	Portable Document Formate
PNG	Portable Network Graphic

REDUCE	REealtime Distributed, Unconstrained Cooperative Editing
RPN	Reverse Polish Notation
RSA	Ronald-Shamir-Adlemann
SEMA	Session Event Management Agent
ShrEdit	Shared Editor
SMS	Short Messaging Service
SVG	Scalable Vector Graphic
U-DL-A	University Digital Library for All
WCWA	Web-based Cooperative Writing Application
WFF/wff	Well-Formed Formula
WWW	World Wide Web
XHTML	Extensible Hyper Text Markup Language
XML	Extensible Markup Language
XPM	X-PixMap

# List of Figures

1.1	Cooperating users in a distributed cooperative environment . . . . .	2
1.2	Web cooperative authoring principle . . . . .	6
1.3	Concurrent authoring based on the document logical structure . . . . .	7
1.4	The PIÑAS architecture . . . . .	9
1.5	Principle of the PIÑAS Event Manager . . . . .	10
2.1	Applications in the Groupware domain . . . . .	19
3.1	Structure of Cooperative Writing Applications . . . . .	32
3.2	Author naming schema . . . . .	34
3.3	Architecture of AllianceWeb . . . . .	34
3.4	Information selection, filtering and presentation . . . . .	39
3.5	The "work focus" and "perception" during synchronous communication . . . . .	42
3.6	MSN messenger notification for starting session of a user . . . . .	45
3.7	The "work focus" and "perception" on a formula based document . . . . .	46
3.8	The "work proximity" notification . . . . .	47
3.9	User authoring actions to start and terminate the cooperative session . . . . .	49
3.10	Session start and document open notification . . . . .	50
3.11	Present activity of coauthor . . . . .	52
3.12	Classification of a document . . . . .	53
3.13	A view of BSCW . . . . .	55
3.14	A view of the EquiText document . . . . .	56
4.1	MathML box and the greek alphabet palette . . . . .	60
4.2	Events produced during the production of a formula . . . . .	62
4.3	Source of MathML representation for the formula $\frac{a}{b}$ . . . . .	62
4.4	Notification: a) well-formed b)not well-formed formulas . . . . .	63
4.5	An incomplete formula: not-well-formed . . . . .	64
4.6	MathML pattern structure tree . . . . .	65
4.7	Notification of available tools . . . . .	77
4.8	The GAIE inference engine informs coauthors about existing tools for tables . . . . .	77
4.9	Handling mathematical expressions . . . . .	78
5.1	MathML patterns . . . . .	83

5.2	Greek letters . . . . .	86
5.3	Observations of the formula size increase . . . . .	90
5.4	Document size increases with time . . . . .	91
5.5	Notification of the presence of experts and their availability . . . . .	92
5.6	The discussion between an expert and a beginner . . . . .	93
5.7	Searching expertise with Expertise Recommender (ER) . . . . .	95



# List of Tables

1.1	Awareness in WCWAs . . . . .	5
2.1	Communication and cooperation . . . . .	28
3.1	Summary of Several Documents . . . . .	54
4.1	Examples of formula transformations . . . . .	75
4.2	Symbols for MathML patterns . . . . .	76
5.1	Complexity of mathematical expressions . . . . .	84
5.2	Elapsed Time to produce MathML expressions (in seconds) . . . . .	89
5.3	Awareness in Expertise Recommender Systems (ERs) . . . . .	96



# Chapter 1

## Introduction

This research focuses on providing dedicated group awareness functions to enhance the common production of users, working on a distributed cooperative environment. These functions are based on Artificial Intelligence principles, specifically those of the knowledge based systems. By means of these functions users can efficiently communicate and coordinate the concurrent activities they perform during a joint collaborative session. Thus, the investigated problem and the provided solutions take place as part of the *group awareness*<sup>1</sup>, a main issue of the *Computer Supported Cooperative Work* (CSCW) research domain. Our approach is validated designing and prototyping in a distributed coauthoring application that allows users to produce shared structured (XHTML, XML) documents, including textual information, figures, video, audio, tables, and multimedia scenarios.

In this introductory chapter, we explain the general motivation that founded our approach in the design and prototyping of group awareness functionalities dedicated to a distributed cooperative authoring environment (section 1.1).

We describe the targeted kind of cooperative environment on which we validate our approach: the cooperative authoring research domain that introduces the principles of AllianceWeb (section 1.2).

We explain the principles of the PIÑAS middleware whose mission is to support the cooperative work on the Internet (section 1.3).

In order to define/support an adaptive distributed cooperative environment, a distributed event management service (DEMS) is designed and developed (section 1.4).

Finally, we define the scope of the proposed solution: we explain the group awareness problematic, presenting some scenarios that highlight the need for a dedicated efficient inference engine (sections 1.5 and 1.6).

We conclude this chapter introducing the main contribution of this research (section 1.7), and the organization of this document thesis (section 1.8).

---

<sup>1</sup>The words written in italic are defined in the glossary.

## 1.1 Motivation

The Computer Supported Cooperative Work, CSCW is related to the technology whose goal is to support groups of people collaborating to reach a common goal or perform a common task [2][63][68]. In order to better coordinate the group activities, the CSCW research domain incorporates knowledge from social sciences to learn more about how people work in group, how the automation affects the behavior of team workers, etc.

CSCW on the Web [13] refers to the collaborative activities performed by a group of people who share computer resources on the Internet to achieve a common predefined objective, e.g. the production of shared documents. As we can observe in Figure 1.1, the collaborators must be considered as distributed over the Internet, working within/from their own private environment. In addition, they may work at different times, and they may simultaneously use cooperative and non-cooperative applications. They frequently move from cooperative (e.g. coauthoring) to non-cooperative activities (e.g. answering to some personal calls, or working on private documents). Likewise, from a private task they may start a collaborative activity. Thus, all actions performed by each coauthor on the shared and/or private resources have to be considered in order to determine the features of his/her presence, work status, and actions.

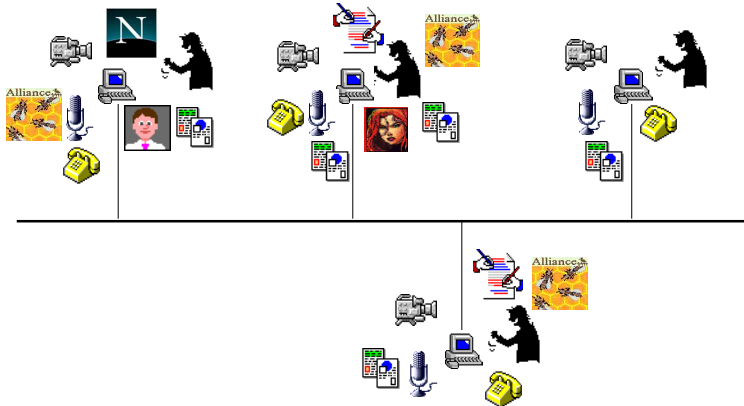


Figure 1.1: Cooperating users in a distributed cooperative environment

Using inadequate and/or limited notification and information services that are basically offered within the cooperative environment, users are generally not able to effectively communicate and coordinate their activities, as well as follow the progression of the modifications concurrently applied on the shared objects. Thus, to support the collaboration, the working environment must be artificially complemented with different communication devices/facilities like telephone, microphone, video conference, instant messaging, that remain external to the cooperative production functionalities. The targeted work consists in providing group awareness functionalities fully integrated to the cooperative production functionalities.

The design of this efficient group awareness system must take into account, how to provide knowledge about presence of users, their activities, and their communication in-

terests. To be able to provide such kinds of knowledge, the targeted system (and then the users) must have precise knowledge of: when did partners start or suspend their cooperative sessions? what did they produce? how much time did the associated effort take them? which are their communication preferences (e.g. synchronous or asynchronous mode)? etc., among other issues.

Providing elaborated, detailed and efficient group awareness notifications, will enable collaborators to accomplish successfully their common task, although it will be necessary to automatically maintain privacy and confidentiality, and to avoid intrusions. Those aspects constitute an unavoidable research issue in the design and implementation of a *groupware* application.

After the study of various kinds of awareness functions provided by cooperative authoring applications (see Chapter 2), we observed a well known lack of *group awareness* functionalities, that was the motivation to design, develop, and integrate a good awareness system within a coauthoring application (see the AllianceWeb editor in section 1.2). The goal of our group awareness system is to adapt and update the cooperative working environment by the analysis of the coauthor authoring actions. We argue that a good intelligent awareness system allows users to pursue successfully their common goal.

Distributed cooperative authoring refers to the activities of a group of people who share computer resources to achieve the production of elaborated shared documents. Working from different distributed Internet sites, users may produce at different times. They may use and combine cooperative as non-cooperative applications, and different communication facilities like telephone, microphone, instant messaging. Their work sessions are highly dynamic and so they may move from cooperative to non-cooperative activities, and vice versa. Therefore, in cooperative environment, coauthors' actions applied on shared resources are important to determine their presence and working status: either performing common activities or being busy in a private and different assignment.

The common understanding (*group awareness*) of the activities performed by all users consists in 1) obtaining the knowledge of each collaborator working context and activities [55], and 2) displaying/integrating them in all user environments in a comprehensive and ergonomic way.

We think that the major features of a *groupware* system consist in the definition of the required common understanding of the shared space:

1. A good communication service to frequently exchange information,
2. Efficient functions to deduce knowledge about the contributions produced by others to coordinate their activities,
3. A "group memory" that records the interaction among the participants and the products developed by them, and
4. Awareness supports dedicated to the current cooperative situation [44], (e.g. copresence notification).

First, we investigate the above needs in the cooperative authoring environment of the **AllianceWeb** [14]. This cooperative authoring application allows authors distributed on the Internet to produce shared complex Web documents [5] in a concurrent, concerted and consistent way. It constitutes the groupware system in which our proposal had been experimented and validated.

Analyzing the awareness functionalities provided by some representative well-known Web-Based Cooperative Writing Applications (WCWAs), we can observe how they provide more or less elaborated functions (see Table 1.1):

- Basic Support for Cooperative Work, BSCW [4][3]. Users are notified about the activities performed by other partners by means of icons. For instance, the creation of an object (adding a new icon), the migration of one object from one location to another (moving an existing icon), the download of a document (download icon), the update of a particular folder document (writing icon).

The awareness functions of BSCW are quite limited, that it does not provide any detail of modifications made on the shared document.

- EquiText [60] [61], by means of buttons, provides information about the document: current and previous version, individual contributions, creation, inclusion, modification, and exclusion of an existing text, as well as the finalization of a document part.

A user does not know that a document is currently being modified by the partners. The collaborators independently produce paragraphs to be integrated to the document. Hence, document consistency cannot be maintained.

- REal-Time Distributed, Unconstrained Cooperative Editing (REDUCE) [76] [75], by means of colors, indicates what part of the document is in process of edition, had been updated or modified, etc.

REDUCE uses algorithms to diffuse changes made by different authors. After the diffusion, no information is available about who has made these modifications.

- The University Digital Library for All (U-DL-A) [18][62] system allows users to annotate documents using different colors showing different kinds of notifications: grammatical errors, syntactical errors, text revisions, demand for a modification directed to the document owner, etc.

Thus, a precise semantic is linked to each annotation: the annotated text is clearly selected by a colored area and the color implicitly provides a semantic information that can be understood by all collaborators at a glance. However, the multiplicity of colors and meanings inevitably becomes a problem.

- AllianceWeb [11] by means of icons associated to each fragment, diffuses and shows the authoring effective role a user acts with. During a cooperative session, each fragment icon is dynamically updated to reflect a special macroscopic event: a new

Table 1.1: Awareness in WCWAs

Writing Applications	Awareness	Coordination
BSCW	Basic icons for document changes (create, download, transfer).	soft lock
EquiText	Basic buttons to inform of previous and current versions, inclusion, modification, and excluding text part.	No cooperative but collaborative
REDUCE	Basic colors to indicate process of edition, updated, and modified part of the document.	Diffusion of produced changes
U-DL-A	Buttons to color fragment allowing to indicate spelling errors, syntactic errors, ask for modifications or to complete a fragment.	No cooperative but collaborative
AllianceWeb	Basic icons delimiting fragments, and showing user role (Manager, Writer, Reader, Null).	New editing opportunities. Fragment updating. Modified fragments.

version of the fragment had been produced, the fragment is free to be open in writing mode, etc. More, these icons constitute a dedicated way to allow users to update the fragment the editor shows within the user’s display.

AllianceWeb offers quite limited basic awareness functionalities, that motivated us to pursue this underlying research.

As we see in Table 1.1, the awareness information provided by these Web-based cooperative writing applications remains quite limited:

*Normally cooperative users can know which specific production has been modified, but they have no precise information about the amount and the nature of the applied modifications.*

Thus, our goal of underlying research was to enhance group awareness within the distributed cooperative authoring application AllianceWeb, in particular:

1. To analyze the activities and working capabilities of users (Chapter 3).
2. To enhance coordination of the collaborators’ activities by automatically deducing ”work proximity” (Chapter 3).

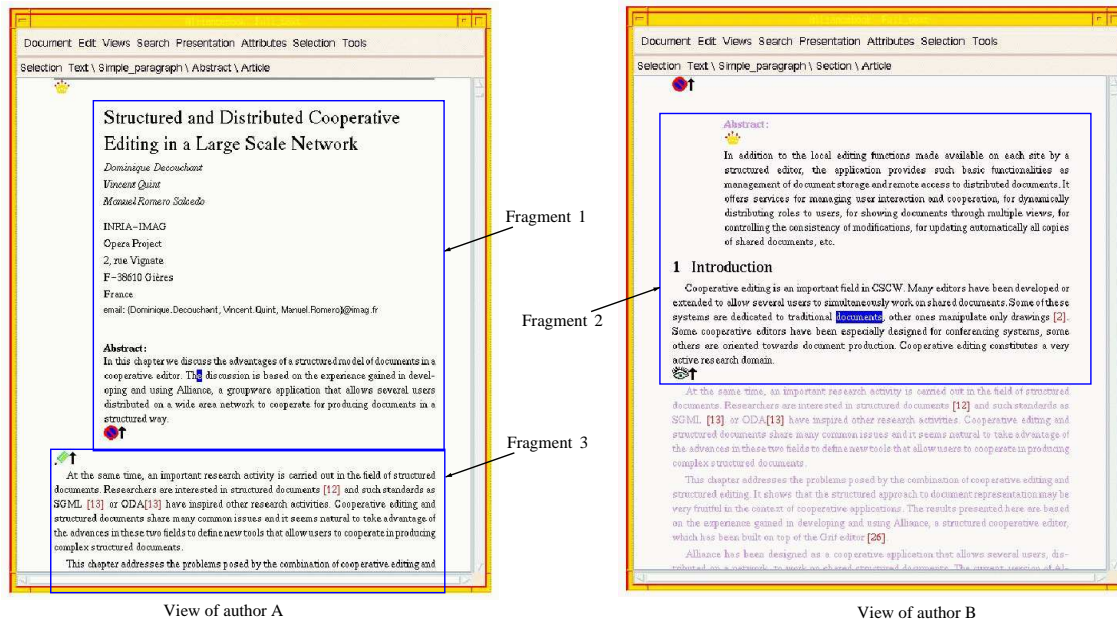


Figure 1.2: Web cooperative authoring principle

3. To allow collaborators in difficulty to contact an expert and to have assistance (Chapter 4 and 5).

In order to validate and test our awareness functionalities, we designed and prototyped a groupware awareness inference engine, which was integrated into the **AllianceWeb/PIÑAS** project, which is explained in the next section.

## 1.2 Cooperative Authoring with AllianceWeb

The AllianceWeb Internet editor allows distributed users to share documents and to perform a variety of cooperative editing actions [13]. This application maintains the efficiency of the coauthors while they produce in a cooperative way, ensures the contribution consistency, and ensures the quality of common production.

Based on the structure of the document (XML<sup>2</sup>) handled by AllianceWeb, a document is partitioned into sharing units called "fragments". Each fragment contains an ordered sequence of XML elements, such as: title, introduction, paragraphs, lists, sections, sub-sections and figures for a chapter document.

Associated to each fragment, roles are assigned to coauthors, to define and organize the cooperative authoring process. Figure 1.2 presents the editing views of a shared document as perceived by authors "A" and "B". This document is partitioned into three fragments. The first one contains: the title, the author names and their affiliations and the first paragraph of abstract. The second fragment consists of the second paragraph of

<sup>2</sup><http://www.w3.org/XML/>



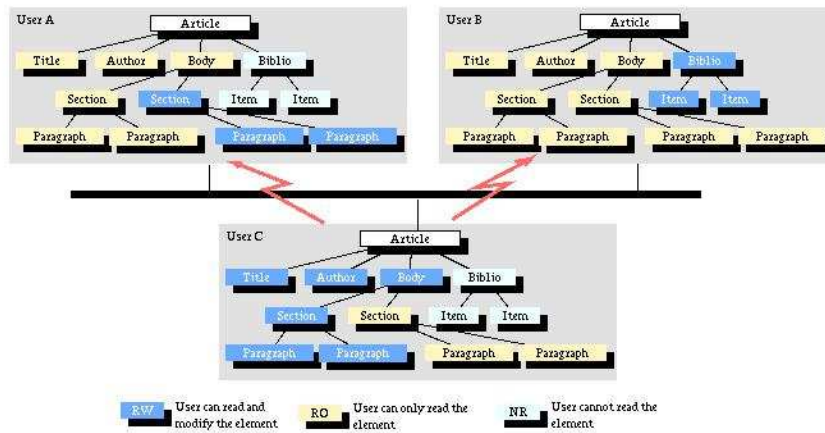


Figure 1.3: Concurrent authoring based on the document logical structure

the abstract, the header and the first paragraph of the introduction. The third fragment contains the remaining part of the document. The first and the third fragments are available to author "A" for writing purpose. More than acting with the writer role, he could also act as a manager. The second fragment is neither assigned to him for editing nor for reading, and thus, he has the "null" role on this part of the shared document.

The coauthor "B" acts as the "writer" of the second fragment. Whereas, third fragment is available to him only for reading purpose (he has the "reader" role on this fragment). The owner (author "A") acts as a "writer" and "manager" having the discretion for assigning or changing the potential authoring roles for the collaborators.

Two authors cannot simultaneously obtain the "writer" role on the same fragment of the document. Hence the "writer" role is mutually exclusively controlled. By contrast, a fragment can be concurrently assigned to more than one coauthor for reading purpose. The document partitioning principle ensures the consistency of the concurrently produced contributions of the coauthors. Thus, as an example, two users are able to concurrently produce on two different paragraphs of the same section. The cooperative application will ensure a consistent integration of the productions in both user environments.

To support the authoring (*fragmentation*, autonomous fragment editing, integration of the productions, etc.) of a shared document, the AllianceWeb editor uses the document logical structure to guide the authoring and managing actions. This structure is mainly hierarchical, complemented by inter-element links (e.g. reference to a section).

The logical structure of a shared document of type<sup>3</sup> "Article" is shown in Figure 1.3. As we can see, coauthors "A", "B", and "C" are cooperatively producing on well identified fragments. Thus, the coauthor "A" acts with the writing role on the fragments whose nodes are shown in blue color: section II, paragraph III and IV. Whereas, at the same time, he can only consult the fragments whose nodes are yellow colored: title, author

<sup>3</sup>Our authoring application allows coauthors to produce: chapter, bibliography, formula, table, report, figure, file, letter, presentation, and structured tree.

name, section I, paragraph I and II. By contrast, the fragments whose nodes are light green colored (item I and II of bibliography) are neither assigned to him for editing nor for reading. Similarly, these fragments are assigned to coauthors "B" and "C" following different authoring modes that respect the main rule of the mutually exclusive productions.

In this way, the AllianceWeb cooperative authoring application follows the principles:

1. **Document partition:** A document is split into several **fragments** following its logical structure,
2. **Assignment of roles:** Each fragment has associated an editing role for each author: [M-Manager, W-Writer, R-Reader, N-Null], and
3. **Mutually exclusive fragment authoring:** The role assignment is dynamic, and the writer role allows to modify a fragment in a exclusive mode.

The AllianceWeb cooperative editor is developed on top of the **PIÑAS** platform [11] that provides a dedicated infrastructure to support Web cooperative authoring.

### 1.3 PIÑAS: a Middleware for Web Cooperative Authoring

The PIÑAS middleware (Platform for Interaction, Naming And Storage) supplies the specific requirements and the dedicated techniques to support Web cooperative authoring environment [11]:

1. User identification: authors are uniquely identified to control the access they perform to the shared resources (e.g. documents) and to protect these ones against unauthorized actions.
2. Naming document and resource: users define from where (storage sites) they can download and upload shared resources and documents, and from where (working sites) they may connect and produce. PIÑAS provides dedicated functions to manage the storage and working sites (e.g. get-document, put-document, locking-fragment).
3. Services: elaborated distributed supports are provided for document replication, and to maintain consistency, and ad-hoc distributed document storage system (see Figure 1.4).

The cooperative authoring actions performed by coauthors on the shared document are captured in the form of events by means of the PIÑAS **Distributed Event Management Service**. This service acts as a medium between the producer (e.g. AllianceWeb editor) and the consumer applications of the events (e.g. a radar view<sup>4</sup> [28] or our adaptive groupware inference engine). One part of our underlying research is related to the design and implementation of this event management service, that is described in the next section.

---

<sup>4</sup>A radar view gives additional information about the workspace where people work and the general structure of their activities.

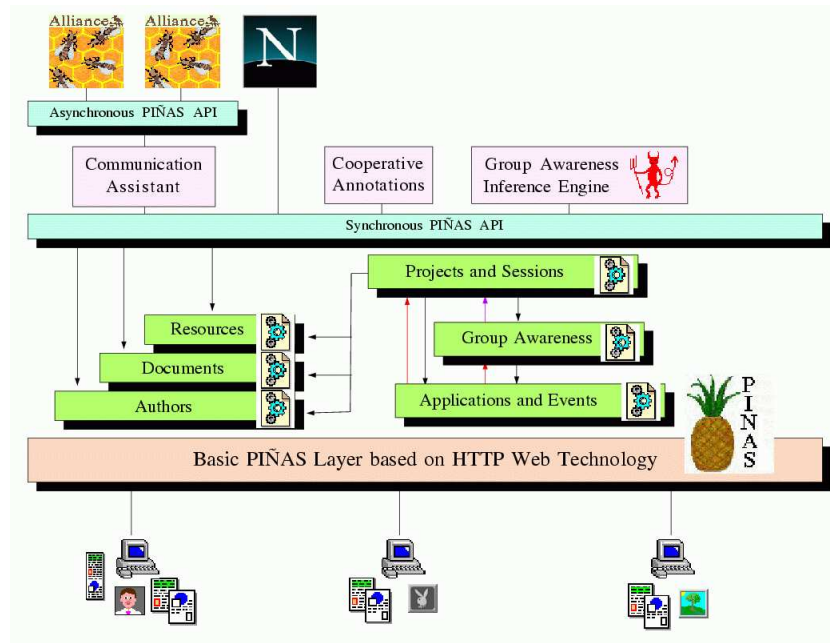


Figure 1.4: The PIÑAS architecture

## 1.4 The Distributed Event Management Service

The goal of the Distributed Event Management Service (DEMS) [12] is to catch/record and manage the actions the coauthors performed on the shared objects. Inside DEMS, these actions are then represented and memorized as events. DEMS proposes functions to client applications to recover and process events. In addition, it provides an easy communication mechanism between cooperative or/and non-cooperative applications.

Events are captured from producer applications (see Figure 1.5), filtered, and then delivered to (eventually distributed) consumer applications. Some of the authoring actions that can be performed on the shared documents are then presented:

- Document handling: document creation, opening of an authoring session, reloading a document (in a cooperatively or non-cooperatively authoring mode), document saving and renaming, etc.
- Authoring operations: selection, deleting, copy and paste of document parts.
- Presentation style modifications: specifying and applying different character sizes, fonts and styles, changing colors, and aligning or formatting document parts.
- Annotation: selection, adding, posting, deleting, and loading annotations<sup>5</sup> linked to some parts of documents.

<sup>5</sup>Annotation is not part of the annotated document, but linked/related to it. Annotation natures can be comments, recommendations, corrections, enquires, etc.

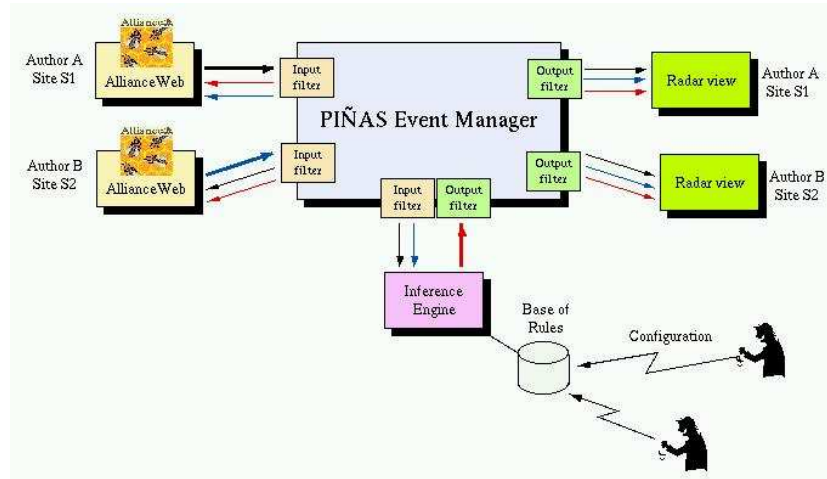


Figure 1.5: Principle of the PIÑAS Event Manager

All authoring actions are captured and treated by DEMS in the following steps: "reception -> input\_filtering -> storing -> output\_filtering -> delivering". All of its services are designed and implemented according to the following principles:

- Events reception and delivery. The producer applications have to send events to DEMS following a "connect-> put-> disconnect" protocol: 1) connection to the DEMS service, 2) storing the produced events to the DEMS's storage service, and 3) disconnection from the DEMS service. Consumer applications follow a similar protocol to invoke the DEMS services following a "connect-> get-> disconnect" protocol: 1) connection to the DEMS service, 2) reading of the events from the DEMS's storage service, and finally 3) disconnection.
- Selection of stored/managed (input and output) events. DEMS controls the incoming and outgoing events. More particularly, events may be blocked or ignored for reception (e.g. repetitive events may be factorized). Similarly, DEMS may prohibit the delivery of some kinds of events to some specified consumer applications. For instance, to maintain the privacy of a user, DEMS may not provide events like "starting application" (session initialization) and "connection" (on line).
- Recording events. DEMS maintains a dedicated storage space to memorize events in order to deliver them to consumer applications. Applications that implement *group awareness* functionalities are typical DEMS clients.
- Identification of the application. Each (event producer and/or consumer) application is uniquely identified by the DEMS in order to control the event diffusions.

More particularly, within the PIÑAS infrastructure, a dedicated consumer application (the GAIE adaptive<sup>6</sup> **Group Awareness Inference**<sup>7</sup> **Engine**) is especially designed to

<sup>6</sup>Ability of the coauthor's environment to adjust itself to the environment changes of another coauthor

<sup>7</sup>Deduction from coauthors' actions or productions to infer relevant group cooperation properties.

capture and analyze the events produced by all cooperative or non cooperative applications in order to synthesize elaborated knowledge that can be provided back to all applications (Chapter 3).

Research work for designing and prototyping a Group Awareness Inference Engine had been practically developed in the field of AllianceWeb, a cooperative authoring application on the Web. This application, developed on the top of the PIÑAS platform (see Figure 1.4), provides fragment based elaborated cooperative authoring features (e.g. fragmentation based on the document structure, user roles, fine grained authoring operations, among others) that allow to follow all user works with a great precision (working focus and the sequence of actions). The idea consists in analyzing the performed actions (nature, order, target element) but also their sequence or repetition, to deduce interesting cooperation properties that will allow to enhance the cooperation process.

## 1.5 Group Awareness Management

In accordance with the document logical structure and the user roles assigned to each document part (fragment), the coauthors need to communicate and coordinate their activities. More particularly, because they are working in a distributed environment, they may not have knowledge about actions performed by their colleagues on the shared document. This kind of common knowledge is called "*group awareness*".

Awareness is then a necessity to support group coordination [55]: being aware of the common production and individual contributions, collaborators become aware of the potential gain of working in group: the members become more efficient than users producing individually. The contributions of coauthors can be highly improved by the understanding of the activities of the whole group. Hence, the group awareness facility appears as an essential functional component of the collaborative applications [58].

More specifically, in order to successfully accomplish a common work, all group members have to be aware of the presence of each other in the shared space, the activities they performed or they are performing, and their communication interests: this kind of awareness is called "presence awareness" [9].

Nowadays, this need for communication and coordination becomes essential for collaborators. Internet instant messaging and presence (IM&P) allow to support some (context independent) communication facilities to allow persons distributed on the Internet, to communicate in a quite easy way.

In this work, we are interested to provide such kind of communication but related to the sharing cooperation environment. By this, we will provide a "contextual" communication/cooperation service.

Events related to partners cannot be considered as the only interesting entities of the cooperative process. Moreover, the group activities performed in the shared space, and the quality and characteristics of common production are also important. This information has to be determined and notified to the interested collaborators: "actions, task, and activity awareness".

Effective group awareness functionalities constitute a major requirement 1) to make users able to accomplish their cooperative work efficiently, and 2) to increase the quality of the common production.

Within the Web cooperative authoring domain, it is necessary to define powerful awareness mechanisms to obtain knowledge about: people, user ability, user actions, user communication interests, group activity, intentions, perspectives, expectations, etc. [31][71][52].

In order to enforce the importance of designing and providing elaborated group awareness functionalities, we briefly introduce two scenarios that highlight some needs for awareness that can be provided to collaborators who are cooperating.

**Scenario 1. Determination of user work proximity** - *As an example, let us consider an author "A" is producing the explanation of a figure while author "B" (who owns the writing role) is modifying the figure content, adding some new elements. Basically, the two authors may be aware of the presence of the other author. However, they did not realize they are independently working on two inter-related document parts.*

This scenario highlights the need to automatically deduce related work focus and coauthors' proximity in order to focalize their attention on this source of possible inconsistency. Thus, the GAIE engine provides automatic functionalities that follow the author actions, deduce such cases of inter-related concurrent modifications to finally notify precise synthesized information to the concerned users.

**Scenario 2. Improvement of inter user assistance** - *To highlight this need, let us examine the example where an author "A" is working on a document table and wants to perform a quite simple editing action (e.g. inserting a row) or a more elaborated one (e.g. splitting a column and organizing some parts in different rows). In a first step, he/she does not want to disturb his/her colleagues to initialize and then achieve his/her task. He/she takes a look at the application menus but he/she does not find any satisfying entry to perform a specific editing action. At this point, it could exist an elaborated (implicitly or explicitly called) tool that will observe all author works and that will classify them as possible "expert", regarding the production (content and efficiency) of the different kinds of document elements. The ultimate goal of this 'intelligent' tool will consist in providing suited tools to "beginners" (e.g. the author "A") allowing them to ask for help to "experts" establishing a focused communication.*

This scenario emphasizes the need to improve the inter user assistance by a) the deduction of working skills and expertise, and b) the notification of available tools that can help users to reach their objectives.

Thus, we implement the following group awareness functionalities into the AlianceWeb/PIÑAS project.

1. *A context-based communication:* The implemented inference engine deduces the cooperative "work focus" (discussion, coordination, production). In addition, each user can know the available, allowed and disallowed communication services of his partners.
2. *Work proximity:* Some shared objects (figures, tables, images, etc.) can be related

with others (e.g. their legend), so they are perceived as an inter-related unit, even though these related objects are produced by different coauthors. Thus, all update actions performed on these objects will be notified to the concerned authors.

3. The evaluation of the cooperative production: We introduce a kind of evaluation for the MathML production (Chapter 5). The objective of this functionality is to provide assistance/help to users, and to announce them, tools to handle MathML [8], SVG graphics [54], and XHTML [35] structures.

We designed and developed an adaptive group awareness **inference engine** to support/provide the above functions, as we see in the next section.

## 1.6 Design and Implementation Principles of the Inference Engine

The adaptive Group Awareness Inference Engine (GAIE) aims at making the cooperative authoring environments more flexible, adaptive, and efficient. The GAIE captures and analyzes produced coauthor's events. Periodically it re-evaluates and updates the shared objects to present them in each author's working environment. The GAIE engine performs the following steps:

1. The retrieval of information from the event repository (knowledge base) about user actions performed on shared objects. This information is handled in the form of events using a dedicated PIÑAS Distributed Event Management Service.
2. The deduction of new facts based on pre-designed rules stored and managed in the Rule Base (RB): for example, to infer properties like the ability/expertise of a user to produce particular document objects (e.g. tables, figures, images, formulas), evaluation of cooperative productions.
3. The proposition/execution of actions whose goal is to provide users with dynamic functions to adapt their cooperative environment: for example, to provide user more efficient tools to handle related or peripheral information.

Thus, the GAIE engine is especially oriented to reach the following objectives:

1. **Establishing a context-based communication.** The "work focus" is dynamically determined to be used in a quasi synchronous communication.
2. **Maintaining the consistency of the cooperative production.** More than one related objects, produced by different coauthors are considered as a unit: "work proximity". All modification of one of them will be reported to the concerned users.

3. **Importing and exporting the preferences of the users' environment.** Given the allowed and disallowed services (e.g. synchronous or asynchronous communication) of the coauthor's environment, the system executes the actions established in rules taking into account the user's preferences.
4. **Evaluating the cooperative production.** The production is evaluated to deduce coauthor's expertise. When this expertise is such an expert, beginners can communicate with him/her to ask for assistance.
5. **Assisting a users in trouble.** For efficient and timely production, the users may ask to be automatically assisted without disturbing their partners.

## 1.7 Main Contribution

The design, development, and integration of a group awareness system to support cooperative productions/interactions, are the main contributions of this thesis. We validate our approach into a Web-based cooperative authoring application: AllianceWeb. We integrate our awareness functions into the AllianceWeb/PINAS project as a seamless system. These functions are developed following Artificial Intelligence principles, particularly those of the Knowledge Based Systems. Thus we designed and implemented an adaptive Group Awareness Inference Engine (GAIE).

The goal of GAIE is to facilitate the *communication* among coauthors, to improve the *coordination* of the *cooperative activities*, and to enhance the common production.

The GAIE engine uses a Rule Base (RB) to dynamically modify or adapt the way by which the group awareness functionality is provided, especially in a distributed environment. The RB is independent of the GAIE inference engine. Thus, depending on the targeted awareness function, it is possible to adapt the RB content by the addition of more rules, removing others that are no longer applicable, and organizing them in different abstract levels. Each rule may be modified/updated by changing its condition part and/or adding new actions.

The kernel of the inference engine remains totally independent of the rule base and more precisely of the different modifications applied to the rules (addition, deletion, and update). Thus, these are some of the advantages of the Rule Based Systems.

The group awareness supports on which this research work focused aims at providing the following functionalities:

1. **A context-based synchronous communication service** [51]: This service provides to the collaborators a precise work focus of discussion. Dynamically, this focus is displayed in all concerned cooperative environments, and it can have a scope that extends to the whole document or only a component of it (formula, table, figure, fragment, etc.).
2. **The deduction of the collaborators' work proximity:** A relation may exist among shared objects (e.g. a figure and its explanation). This relation is dynamically



determined when one of them refers the other one. If one of the related objects is modified, and because these objects are considered as a whole or better as an inter-related unit, the collaborators are notified.

The notification of the work proximity of collaborators ensures cooperative production consistency. The coauthors have the possibility to coordinate themselves and to negotiate the changes concurrently produced on the related objects.

3. **The evaluation of the cooperative production:** The evaluation refers to the analysis concerning: a) the kind of elements included in the shared document: text, tables, links, MathML, SVG schemas; b) the number and the complexity of these elements, and c) the elapsed time to produce them.

The evaluation of cooperative production is used to classify an author as a possible expert. Thus, we attempt to evaluate users skill. In case of production troubles (inconsistencies, errors, etc.), the author is notified if an expert is currently working or he/she is informed about the available suited automatically facilities offered within the authoring environment (handling Math expressions, SVG graphics, tables, and figures).

4. **An assistant for handling MathML formulas:** An assistant helps users during the production of mathematical statements: verifying if the formula is well-formed, reusing a formula by transforming a selected pattern to another one, rewriting a formula into prefix and postfix forms.

We argue that to improve *coordination*, and to make cooperative work more efficient, it is necessary to maintain a high level document consistency, as well as to try to avoid possible conflicts among collaborators.

## 1.8 Thesis Overview

The thesis is composed of six chapters and four appendixes, including conclusions and results obtained from this research work.

**Chapter 2** introduces the research domains this work is related to: the Computer Support Cooperative Work (CSCW), Groupware, and the Group Awareness functionality.

**Chapter 3** describes the principles and the targeted objectives of the inference engine whose main goal is to provide group awareness support between collaborators. The PINAS architecture is introduced, including the Distributed Event Management Service (DEMS) and the AllianceWeb editor. This chapter also presents the obtained significant results of the studies developed to enhance *group awareness*. We compare our approach with some related research works.

**Chapter 4** is devoted to describe the assistance provided to users during their production. We study and analyze several MathML patterns to evaluate if a formula is well-formed, and we develop a tool to allow users to transform a formula into another. More,

users are assisted by notifying them the key shortcuts that are not explicitly available in the dialog interface (menus and tools bar) of the authoring application.

**Chapter 5** is devoted to the group activity coordination. The production of a user is analyzed to deduce his/her expertise for writing complex and structured Math expressions. Hence, we establish some criteria to evaluate the complexity of expressions. Likewise we summarize the different components of the production: elements, attributes, presentation, XHTML, MathML, SVG schemas, etc. Using the evaluation of the production, the author is classified as a possible "expert". This chapter presents the utility of a dedicated assistant to help coauthors to enhance the production.

**Chapter 6** concludes this dissertation describing the main contribution of our approach.

# Chapter 2

## Related Domains to Workgroup Computing

Our research is related to the domain of Computer Supported Cooperative Work (CSCW) (section 2.1), the Groupware (section 2.2) and the Group Awareness (section 2.3). There is a twofold tradeoff between awareness and privacy, as well as between awareness and intrusiveness (section 2.4). In a groupware system, the communication services offered to users (section 2.5) and the tools for coordinate the users's activities (section 2.6) are closely interrelated to the group awareness.

### 2.1 Computer Supported Cooperative Work

Computer Supported Cooperative Work, CSCW, is an interdisciplinary research domain of the engineering field and the social sciences [25]. The engineering discipline deals with the design and the development of suitable computer systems to support the work produced by a group of individuals and organizations. This is the Computer Supported, CSinterpretation of the CSCW term. The support can be performed into two levels of abstraction: a) the content level: focuses on the creation and manipulation of the structure of the shared information. For instance, databases and knowledge bases are characterized by a high content orientation, and b) the process level: focuses on the information production process. For instance, e-mails and video conference services.

Giving emphasis on the CS interpretation, CSCW is defined as " ... *the design of computer based technologies with explicit concern for the socially organized practices of their intended users*" [63].

In order to better organize the group activities, it is required to integrate knowledge of sociology, psychology, anthropology, telecommunications, organizational theory, and information management. This is the Cooperative Work, CW interpretation. This interpretation can be diverse: a) this is a work process where several persons are involved, b) in cooperative work, the members perform activities to obtain a mutually decided predefined objective, c) the cooperative work is a special type of work without hierarchical structures

and with a high degree of autonomy of the individuals to produce.

The CW interpretation enforces social factors of the group members, for instance: members want to maintain certain level of privacy and confidentiality regarding their production. At one moment a member has the interest to communicate with others, while at another moment he/she wants to concentrate on his/her production and he/she does not want to be disturbed (availability). In addition, the production time zone (e.g. Europe-time zone is different of America time) must be considered, and the assignment of roles should be regulated.

Thus, when the importance is given to the social issues, CSCW is defined as, " ... an endeavor to understand the nature and characteristics of **cooperative work** with the objective of designing adequate computer-based technologies" [2].

In order to equally integrate technical engineering facilities (CS interpretation) and social factors of the group (CW interpretation) in a multidisciplinary and a multiregional organization, CSCW is defined as, "... a generic term which combines the understanding of the way people work in groups with the enabling technologies of computer networking, and associated hardware, software, services and techniques" [73].

In general, a computer system that allows users to reach a common objective is denominated a **groupware**. The next section deals with this field.

## 2.2 Groupware

Groupware is defined as, "the computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment" [15]. A groupware system can involve software, hardware, services, and group process support [38].

The principal features of a groupware system are: synchronous or asynchronous concurrent production, group awareness, control of production, coordination of users' activities, and identical or different perception of shared environment.

In Figure 2.1 we can see some groupware applications characterized by two parameters: place and time. As we can observe, the cooperative editing can be performed in all cases: in synchronous or asynchronous mode, and distributed or local processed information.

From the groupware applications shown in figure Figure 2.1, we only talk about the cooperative editing

**Cooperative editing:** Users can jointly compose and edit a shared document. In the asynchronous mode, a user produces and uploads his/her production to the assigned storage site, later it will be available for his/her colleagues. For example, BSCW [4][3][1]. In the synchronous mode, users can produce at the same time and perceive all the cooperative updates at the same moment. For instance, REDUCE [76][75], GROVE (GRoup Online Viewing Editor) [15]. The **GROVE** system organizes several group activities like joint papers, presentations, and brainstorming.

The developers of a groupware application must take into account: a) cost and benefit, b) integration of several services (e.g. communication, annotation, version tracking, dis-

	Same place	Different places
Same time	Decision Support Systems Electronic Meeting Room Shared WhiteBoard Computer Aided Education Computer Aided Design Cooperative Editing	Shared Office (MediaSpace) Video Conferencing Computer Aided Education Cooperative Editing
Different times	Project Management Shared Office System Computer Aided Design Cooperative Editing	Electronic Mail Workflow Systems Distributed Diary System Cooperative Editing

Figure 2.1: Applications in the Groupware domain

tribution of drafts), c) social, political, and motivational factors (e.g. priorities, interests, occupations, roles), etc. [23]. In addition to these considerations, a groupware application must integrate functionalities to inform its users about different happenings in the environment. In order to be conscious of this information, these kind of functions concern the **awareness**. This concept is treated in the next section.

## 2.3 Awareness

What does "Awareness" mean?, the dictionary definition says: *"having knowledge of", "consciousness of", "state of elementary or undifferentiated consciousness"*. Whereas "knowledge" means: state or fact of knowing, familiarity, understanding, learning, gained information through experience or study, specific information about some happenings.

The classic definition of awareness in cooperative work is: *"an understanding of the activities of others which provides a context for your own activity"* [55].

The group members working in a shared environment need to know/be aware about:

1. **the working requirements:** Information about the shared objects, environment, and cooperative/collaborative activities. For instance: Who is currently present in the shared environment? Who is active and who is inactive? With whom communication can be established? Who is expert? Who is available to ask for help? When is an action performed on a shared object? To whom and for what purpose should information be notified?
2. **the processing solution:** The knowledge about how information can be gathered and transmitted, how a result is obtained or a goal is achieved, by which means an action is performed.

We study the different awareness functions provided by different groupware applications (computer aided education, cooperative authoring, developing software) and according to the two above points, awareness can be diverse, as we will see in the following subsections.

### 2.3.1 Group Awareness

Group awareness is related to the information about the state and activities of work team: presence, executed activities, and the location from where they participate [27]. A classical definition of group awareness is: *"the understanding of who is working with you, what they are doing and how your own actions interact with their ones"* [56]. The first system to develop this kind of awareness is **Portholes** [56].

The Portholes project supports awareness functions for distributed working groups. It provides a daily work environment view. This information is transmitted through images captured at various locations where group activities are performed (offices, laboratories, public spaces, etc.). The captured images are explained by textual information and are stored in a database. The group members connected via Internet can retrieve these images to know the working states of their colleagues. The awareness functions in this system are related to: "knowledge about who is acting around, what activities are performed, who is talking with whom, ...".

The asynchronous Collaborative Writing Website, **CoWeb** [30], is another system which offers: Monitoring features, navigation support, re-classifying material, notification and annotation support, templates and customization support, programming interfaces, and on-line administration utilities.

The working requirements for group awareness are addressed to particular roles in the cooperative learning domain: authors, purpose agents, central users, peripheral users, site designers, developers, administrators, and support staff.

The CoWeb awareness support provides the date and time of user actions performed via e-mail: creation, modification, opening, and annotation (users are invited to comment CoWeb pages).

The REal-time Distributed Unconstrained Cooperative Editing, REDUCE working on the Internet [75][76], uses a "**top-down**" approach to determine user's needs and identify relevant group awareness mechanisms in order to be implemented.

**REDUCE** is used in a laboratory-based usability experiment [69], where users can produce documents (e.g. creative writing, technical document preparation, and brainstorming).

In order to provide group awareness in this experimentation, the following components are integrated into the REDUCE system:

- Modification Director(MD): It notifies by a flashing icon when user's contribution is modified by others. Clicking on this icon, the concerned modified section is displayed and reports who made this change.

- Toggle Multi-user Scrollbar (TMS): It displays the position of a user within the document. Users can hide or show remote user's scrollbars. The TMS displays the remote user's name near to their scrollbars allowing others to know who is producing.
- Split Window View(SWV): It displays view of users and their working sections. Remote insertion cursors are also added to display the user's working areas.
- Dynamic Task List(DTL): The task-based awareness presents an active and frequently-updated list of all performed tasks indicating user's name and text color identification.

The "top-down" approach shows the important information which should be provided to collaborators: information about user's responsibilities, their current work areas, their current actions, etc. Such information enhances group awareness.

### 2.3.2 Action, Task, and Activity Awareness

The information related to user's actions, common tasks, and activities refer to this kind of awareness: Who did perform an action (identity)? From where (location)? Is it better to process the specification and execution of the activities of a common task, with a cooperative or non-cooperative application? The awareness provided by BSCW [3][4][1] is an example of this kind of system.

**BSCW** cooperative editor notifies through icons the activity executed by users. For example, a "New" icon shows the creation of a document, a "Hand" icon signals that something has happened on an object inside a folder, a "Spyglass" icon indicates that an object has been downloaded by someone, and a "Pen" icon represents some modifications on an object such as edited, renamed, or changed description. Clicking on any of this icon displays a list of events related to the icon, describing what was done, when and by whom.

On the other hand, the knowledge that makes people to carry on coherent discussion, to observe and imitate other's actions, to engage in peer pressure, to create, to notice and conform to social conventions, are the collective awareness, as it is satisfied in Babble [16].

**Babble** supports communication and collaboration of small groups (30-40 members). The group members have information about how many users are present (availability) and from them how many users are active or inactive in the shared space (represented as a social proxy in the form of a circle), and who is listening or being silent for a certain time. The members currently present in the session and active in conversation (writing messages or listening one), are shown inside of a circle in colored dots (each user has its own colored dot). An inactive user is represented by a dot that moves toward the edge of the circle.

### 2.3.3 Workspace Awareness

The workspace awareness provides information at a glance about user's activities, and fine details of the performed actions on the shared objects. Some elements considered into the workspace awareness are: actions, tasks, intentions, changes, objects, extents,

abilities, spheres of influence, and expectations. For instance, when does someone start the cooperative session? when does a user leave the session? what does a user perform? and what object does a user process?

Workspace awareness can be defined as: *"the collection of up-to-the-minute knowledge a person holds about the state of another's interaction with the workspace"* [28].

The real-time distributed groupware [26][28][29] analyzes some workspace awareness **"widgets"** in the relaxed-**WYSIWIS** (What You See Is What I See) groupware [67]. The concerned awareness to this kind of system is the awareness of identity, location, and actions. This information helps people to maintain the track of others, because in this kind of system people can change the location or representation of their view onto the workspace to suit the requirements of the immediate task.

In a relaxed-WYSIWIS workspace four inventions help people to keep track of others:

- Radar views: miniature overview of an entire workspace. The radar display shows the extent of what each person can see by marking view outlines and also shows finer-grained location by including miniature telepointers that represent each person's mouse cursor. The view outlines of each member is shown by a unique color;
- Multiple-WYSIWIS views show a scaled-down duplicate of each person's view of the workspace including cursor movement and manipulation of artifacts;
- What You See Is What I Do, WYSIWID display shows the detail of actions performed by a user; and
- Workspace teleportal allows people, without much effort, to 'glance' at another's work area in full size view.

### 2.3.4 Peripheral Awareness

The information about activities being executed in the periphery, independent of the user attention, concerns to the peripheral awareness. Systems that offer this kind of awareness are: **AROMA** [57] and **SideShow** [7].

Abstract Representation Of presence supporting Mutual Awareness, AROMA, provides abstract information about who is present in the periphery and where they make changes [57]. The abstract representation of information collected by audio-video devices (e.g. sound level, user identification) are then synthesized by output devices (e.g. screen, audio detectors), and transmitted to the required sites. The audio signal is picked up by an output device and used to determine the number of people in a location.

SideShow provides awareness via a sidebar on the primary display of users. The sidebar contains a number of items called "tickets". Each "ticket" provides information about the user's activities, availability and unavailability, and messages received. When a user works on the computer, the SideShow system periodically collects information from its peripherals.

### 2.3.5 Contextual Awareness

This is the information concerning working states of users, to adapt their current situation. For instance, consider a shared workspace system used by several hundreds of people, and containing several thousands of computer artefacts. In this case, it is better to provide



information like identification of people, their responsibilities and their needs, than to provide the complete information about each participant and artifacts. **POLITeam** [45] provides this kind of awareness.

Telecooperation by Electronic Work Process, **POLITeam** uses some conventions for group members like: naming documents (e.g. by producer, semantic, content, technicality, requirement), sharing task processes (e.g. authority, responsibility, right, producing new document and assigning document type), and executing roles (e.g who is responsible, when a user is absent). Hence, the corresponding information is provided to a group member according to his work situation. For instance, when a user opens a document, the related information is: who else has used it? and what was the production effect?

### 2.3.6 Conversational Awareness

The information about states of communication of group members, i.e whether they are listening, hearing or writing, refers to the conversational awareness. **GAZE** offers this kind of awareness functions [71].

**GAZE** captures, conveys, and renders information about the human intentions: a) information is collected by video cameras, microphones, manual input devices (like mouse, keyboard). b) from eye and head movements, users know who is communicating with whom. When a partner manipulates a manual device, users know who is working on a document and ask others to see it. c) using eye and head tracking devices. **GAZE** provides a 2D images of participants taking part in a conversation. Thus, distributed users are aware of conversational intentions of their colleagues.

### 2.3.7 People, Document and Collection Awareness

*"The knowledge of who is on-line, who are at the same instant visiting a virtual place like a chat room, or web page, and who access the same document at different time and from different locations. Generalization of these knowledge is collection awareness".* **Livemaps** (chatting and collective surfing) provides this kind of awareness [10].

**Livemaps** displays a visual representation of the structure of a collection of Web pages, and provides a graphical representation of the site map, the page's nodes are colored to show the presence of users and the accessed document. Two roles are distinguished: the visitor who surfs the cyberspace, and the observer (webmaster) who observes surfers visiting a particular site. All visitor actions are monitored when a user accesses the site: what documents or pages are opened, who access the same documents. This information may be notified like Instant Messaging service does [21].

### 2.3.8 Perspective Awareness

When the group members work at remote locations, the communication gap among them increases so the perspective awareness is necessary to better coordinate activities of the group and anticipate intentions of collaborators to do next (perspective) [36].

**TeamSCOPE** [36] offers complete information about past, current, and potential activities of team members.

The following components are part of the awareness functions:

- File Manager keeps the history of who else has accessed a particular file or folder.
- Message Board posts a message that can be latterly responded by a team member.
- Calender indicates special events and their dates.
- Activity Summary provides information about activities of team members.
- Activity Notifications sends activity information to users via e-mail.
- Team Member Login Status shows who else is present or has accessed the document in the past.
- Team Member Usage Information notifies members about the access done to a particular page by different team members.
- Team Summary Site provides a quick summary showing accesses to a file, messages, and calender entries.

### 2.3.9 Synchronous/Asynchronous Awareness

In some cases the time to deliver information becomes highly important, when the information is delivered at the same time that it is produced, the awareness functions operate on a synchronous mode, otherwise asynchronous. The Information may be about current and past collaborative activities, and the notion of place rather than session or meeting [65].

The **I2I** system [6] offers synchronous awareness. I2I interchanges information among several processes, about user actions and document content. I2I is based on Microsoft object-oriented inter-process communication service. A central broker maintains information about users profile (name, passwords), the locations from where documents are accessed, documents currently updated, significantly modified documents, new created documents, etc. In addition, users can communicate among them, when they access the same document.

Not all group awareness systems offer the awareness functions in synchronous mode. For instance, Portholes [56], CoWeb [30], TeamSCOPE [36], etc.

### 2.3.10 Presence/Social Awareness

The information about the presence of group members in the shared workspace, their motivational state or attitude during a communication session, and the performed actions on

the objects, are related to the Presence/Social awareness. Implicitly, this kind of awareness includes the action and activity awareness (section 2.3.2). The awareness provided by **Virtual School** [9] is representative of this kind.

The "science notebook" is a main component of the collaborative editor Virtual School. Student users can build, organize and share projects and experiments, with "science notebook" component. The "session manager" component provides information about the current activity of users, their affiliations, and location. Additionally, users can "annotate" any collaborative production using the "science notebook". During the edition action, the status of the "page" is represented by colors: pages currently edited are shown in green color; "pages" in reading status are colored in yellow. To coordinate their activities, student users have the possibility to use some communication services: video conference, text chat, and e-mail.

## 2.4 Privacy, Intrusiveness, Security, and Awareness

A groupware must integrate a well designed awareness system, taking into account the privacy requirements of users, minimal intrusiveness to user work state, and the security of transmitted information.

**Privacy** involves the ability for individuals, small groups, and organizations to negotiate mutual demands on time, space and actions [40]. Thus, in our context, privacy is related to the knowledge about the workplace (from where user produces) and the user's workspace (activities or tasks to be accomplished).

The BSCW, cooperative editing application, provides an **access controlled model** [1] to ensure privacy. Giving a particular event, the information is filtered according to the role and requirements of users.

The Instant Messaging systems (e.g. MSN Messenger<sup>TM</sup>) maintain user's privacy. When a user does not want to communicate with his/her colleagues, he/she can change to "off-line" his/her login status. He/She can see who are on-line, but he/she is not visible to them. Thus, no one can start a chat session with him/her.

Participants must feel free to exchange information with any partner, but this exchange should not disrupt the activities of other members. Only the necessary information should be filtered and transmitted as well as this one must not be intrusive to the user's work state.

An awareness system must maintain an equilibrium with privacy [33], for instance:

1. the shadow-view technique: To provide information about the movement of people by a static reference image. The images are darkened in the area where the people are currently active.
2. the shared audio technique: To process a speech signal into a non-speech audio signal that has several critical properties, e.g. removing intelligible words, reducing volume, etc. This technique allows the recognition of a speaking user without knowing the matter of talk.

3. the synthetic group-photo: To focus on information about the presence and absence of colleagues by means of an individual and aggregated group photo. The inclusion and omission of one's photo allow users to know who is present and who is absent.

**Intrusiveness** concerns with the notification of information regarding group activities that not manifest interest or benefit to whom it is provided.

Unwanted information produce adversely effect, since users loose their concentration on the task carried out.

An awareness system must provide the means to control the intrusiveness [21], for instance: - to verify who has the right to see or hear users, when someone is seeing or hearing someone: - to determine what intention is behind a mediaspace connection; - to authenticate the connection; - to determine when certain information is needed by a group member; - to enable participants to control exactly who is allowed to display, see, and listen them; etc.

Sometimes, information may be used incorrectly, for instance, secrets about the organization can be transmitted to working team, who must not know it. Thus, measures should be taken to secure the information while it is transferred from one user to another.

**Security** concerns with the protection of information of unauthorized access. One way to secure information is to encrypt and decrypt it.

An interface designed for media space conversations [66] secures the user interaction by the Digital Encryption Standard (DES). For instance, a collaborator sends RSA public key (Ronald Shamir Adlemaan) to other user, with whom he/she wants to communicate. Once the latter can decrypt the encrypted received data (using the public key) they can interact.

An awareness system must ensure the security of the information handled and the privacy of users, as well as not be intrusive in the presentation of information to the users.

## 2.5 Communication and Awareness Support

Communication is a discipline in charge of conveying information or ideas by speaking, signaling or writing, between people or groups. People working in groups require some service to transmit information or to make different queries, using several techniques like highly interactive face-to-face meetings, or less interactive text-based exchange of messages as postal communication and noticeboards. Both active and interactive kinds of communication have been included in the **Computer Mediated Communication** (CMC).

CMC covers the concept designed and developed in the 1970s, i.e. *"the use of the processing and storage capabilities of computer networks to support communication processes within a group, as well as tailor those processes to the requirements of the application and the nature of the group"* [70]. A representative CMC system is Emergency Management Information System And Reference Index, **EMISARI** [32].

The following characteristics of EMISARI are not found in most current "groupware" systems:

1. Quantitative communication structures. People communicate not only with words, but with numbers. This communication is called "computer conference". A monitor of this conference could establish data structures that were single numeric values, columns (vectors), and tables (matrices). A single member of the conference could be made responsible for reporting the specific numeric values, vectors, or tables and he/she could build vectors, and matrices from the numbers sent to him/her. The users could also report when something is wrong (e.g. not yet available data).
2. Content-based communication. In this communication, the content of the communication item determines the "address" or the ultimate recipient to establish the communication. A user can send a message to an individual data item, and the concerned users will be notified of any change as well as, new comments about it. Thus, the single data item causes a communication thread among users.
3. Indirect communication. This communication is established by the system, but based on the user's actions. The users search items in the notebook which is used for policy interpretations. All members can read this notebook, but only a specified subset of members can write on it. The computer would make available to the people to write into the notebook the list of words that people were searching for but "not found". The indirect communication channel allows the writers to determine what was missing from the material and could be used to schedule the discussion according to new interpretations.
4. Roles. Behavioral expectations that individuals take on with the group of which they are part. For example, the system supports a role of an instructor and a student to be played in the communication systems specially designed for remote education.
5. Notifications. The system messages dedicated to informing people of what they need to know as a result of action of other members of the group. For example, in educational systems, students are notified when a new grade set is entered, thereby eliminating the unnecessary inquiries.

CMC can be synchronous or asynchronous. When people interact at the same time, the communication is **synchronous**. For instance, audio and video conference offered by the GAZE system [71], instant messaging, and chat provided by the Virtual School system [9].

Sometimes different time zones of the member's work locations can be a primary drawback of this communication.

When the interaction is at different time, the communication is **asynchronous**. For instance, e-mail, streaming (audio and video), Web logs, message posting. The systems like CoWeb [30] and SideShow [7] offer this kind of communication.

The asynchronous communication is useful for postponing a dialogue, allowing people to communicate from different time zones, and facilitating users to attach documents. But, this form of interaction may cause unwanted delay in the completion of joint tasks.

Table 2.1: Communication and cooperation

Kind of communication	Cooperation
No communication	Decreased
Text chat	Increased
TTS and voice	Enhanced

In addition, the written message may contain some unclear points that need more explanations. However, both synchronous and/or asynchronous communication are useful to coordinate user's activities.

The implementation of an appropriate kind of communication is guided by the work domain and the cooperative application used by members.

The communication effect produced on the cooperation was tested in [37], that provides four kinds of communication among people who were in cooperation and in competition game: no communication, text-chat, text-to-speech (TTS), and voice (e.g. speakerphone). The results are shown in the Table 2.1. The experimentation design took into account the task performance and user preferences for cooperation. It is proved that the voice and text-to-speech communication had a greater impact on the level of cooperation.

The communication service allows group members to coordinate their activities.

## 2.6 Coordination and Awareness Support

In the cooperative/collaborative production process, group members perform interdependent activities using some shared resources. Conflicts may arise among member's activities due to limited number of shared resources, roles, priorities of tasks, and interests of users. The management and regulation of these conflicts is termed as coordination.

So, in general terms, coordination is "*managing dependencies between activities*" [43]. The coordination processes manage the scheduling and synchronization to solve constraints. For instance, when the dependence concerns with the "shared resources" or task assignments, the coordination process can select one of the following criterion: "first come/first serve", priority order, budgets, managerial decision, market-like bidding, etc.

In the context of CSCW, the coordination is defined as "*the support for the activity of managing dependencies and possible conflicts between collaborative entities (users and their roles) involved in common and inter-related tasks of a collaborative activity (actions performed in the shared workspace)*" [31].

In [31] a flexible model was proposed to provide the coordination support. This model is adequate for collaborative tasks developed in the context of workgroup activities. The model determines the structural elements of the organization and the activities performed by these elements (dependencies). The identified elements are: people, roles, activities, processes, shared information space, workspace, activity space and process space. When activities are in conflict, the coordination process notifies this fact to the related actors by

means of a **Session Events Notification Service**.

The Session Events Notifications Service is developed on the Dágora (Distributed ágora) platform: a CSCW platform for the support of flexible and scalable groupware applications. The notification service follows a publishing/subscription methodology acting as a tailored component between event suppliers (the collaborative applications) and event consumers (session notification objects). The session notification provides:

- Short Messaging Service (SMS) that notifies events on user's pagers or cellular phones.
- Mail: events are notified as E-Mail structured messages.
- User Event Queues. Each session maintains User Event Queues where events are logged. The events are captured asynchronously on the user's request. A POP protocol (similar to POP mail protocol) is used to capture events from event-queues.
- Session Event Management Agent (SEMA) to notify events synchronously when users are running an instance of the session manager, when they are "logged" in a session.

Another well-known groupware system, the "**coordinator**" system [19] was developed to act in accordance with the linguistic terms. "*The coordinator is a system for managing action in time, grounded in a theory of linguistic commitment and completion of conservation*". Users can select menus to make requests, promises, and to start conversation. Likewise, they can respond the messages selecting the "Answer" menu. The "coordinator" also provides e-mail service. People coordinate their activities by anticipating the behavior of others through a language: to make requests or promise to perform some actions.

Robust group awareness functionalities, a communication service, and a coordination support become integral components of a groupware system.

## 2.7 Conclusions

A groupware application must take into account the social requirements of group members: When they want to be informed about activities of their colleagues? When they want their activities may be notified to others? When they have the interest to communicate with? When they want to be informed their activities are interdependent? When they want to maintain privacy? etc. All these requirements can be satisfied by means of an efficient awareness system. This awareness system may be composed of a communication service, a coordination support, and a notification mechanism.

In the next chapter, we present our approach to enhance the group awareness functionality in the AllianceWeb/PIÑAS project.





## Chapter 3

# A Group Awareness Inference Engine to Enhance Cooperative Work

This chapter focuses on the design and the implementation principles of the Group Awareness Inference Engine (GAIE), which aims to enhance **awareness** among coauthors in order to produce shared documents in a consistent and well-informed way. The proposed GAIE inference engine is based on the following principles: 1) information catching, 2) deduction of new facts and 3) proposition/execution of actions. These principles aim at retrieving information about user actions performed on shared objects. The information is handled in the form of events by the PINAS Distributed Event Management Service (DEMS).

The GAIE inference engine is integrated into the AllianceWeb authoring application (section 3.2). Firstly, to clearly introduce the context of this study, the structure of cooperative writing applications is briefly presented and explained (section 3.1). The observed and/or treated facts are deduced from the handled shared entities and the information is presented to each interested user in order to adapt his/her environment (section 3.3). Thus, the goal of the inference engine consists in detecting, inferring, and notifying collaborators of their partners activities in order to enhance the group awareness (section 3.4).

In order to achieve this goal, the GAIE inference engine deduces the "work focus" of each coauthor and transmits it to his/her colleagues enabling them to coordinate their activities (section 3.5.1).

Within the framework of the structured shared production, the inference engine determines the eventual proximity among coauthors and minimizes the possible conflicts that can emerge (section 3.5.2).

Additionally, the GAIE inference engine notifies a user about the authoring activities of his/her colleagues: author's session opening and closing, presence or absence of a coauthor, or the evolution of the content of the shared document (sections 3.5.3 to 3.5.6)

We conclude this chapter presenting our contributions to solve the problem whose goal is to define and prototype group awareness facilities for Web-based cooperative editing (section 3.6).

### 3.1 The Structure of a Cooperative Writing Application

The typical architecture of a cooperative writing applications, like AllianceWeb, can be decomposed in three different functional levels: a) the information transport layer, b) the document support layers (e.g. PINAS platform), and c) the application layers (see Figure 3.1).

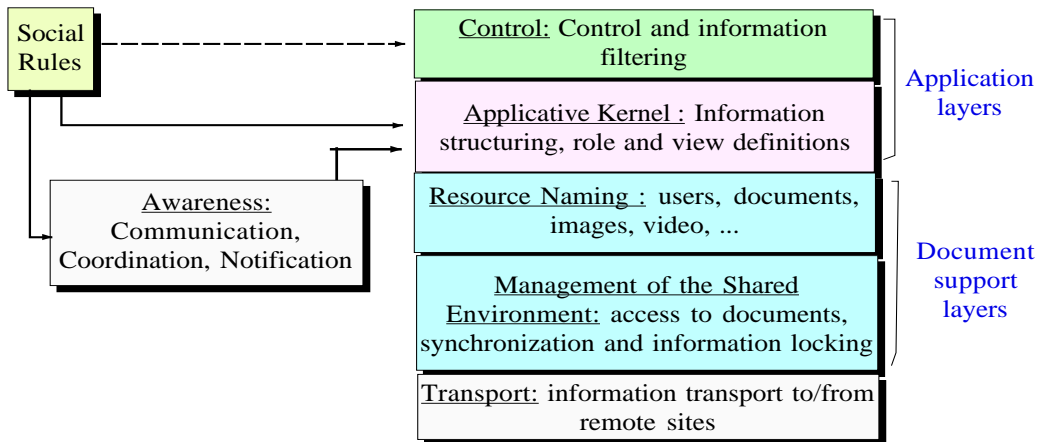


Figure 3.1: Structure of Cooperative Writing Applications

**A) The Information transport layer.** In order to support browsing, editing, and Web publishing, the information transport layer is necessarily based on the HTTP<sup>1</sup> technology. The HyperText Transfer Protocol establishes a stateless connection between a client and the remote Web server processes. Thus, during the browsing process, the client accesses are generally anonymous despite the fact that "firewall" mechanisms can be installed on some Web servers. Web editors, like Amaya [24], require to well identify the client site and/or users. This identification process is used to verify if a particular user is authorized to consult (HTTP Get operation) or modify (HTTP Put operation) a particular document.

**B) The Document support layers.** In order to cooperatively produce, an interaction model is required to manage the shared environment, as well as a way to identify all partners, the results to be obtained, and the resources involved in the cooperative process. Thus, a dedicated naming system is required to define/designate users (coauthors), documents and resources. This naming system constitutes the base to control the accesses to shared documents and the roles users may act with.

The AllianceWeb user, document, and resource naming system is based on the Uniform

<sup>1</sup><http://www.w3.org/Protocols/>

Resource Locator, URL<sup>2</sup> and uses the Common Gateway Interface (CGI)<sup>3</sup> script facilities. A CGI script is a standard program that may be called to perform specific actions on the Web server resources, like XHTML<sup>4</sup> documents [35], or other kind of resources (files, processes, databases, etc.).

**C) The Application layers.** In this layer, we distinguish the kernel of the cooperative application, as well as the control and the information filtering. One of the most important characteristics of a cooperative application is the structuring of the shared environment. Thus, data structuring is used to define the sharing units on which collaborators may act in respect with their assigned roles. The cooperative application offers functions by which each collaborator has a global perception of the production progress.

## 3.2 The Architecture of AllianceWeb

AllianceWeb uses the World Wide Web (WWW) technology to support accesses to remote information and to manage the distributed or replicated information [11]: the HTTP protocol, the URL based naming system, and the CGI scripts.

The naming system used by AllianceWeb identifies users, documents and resources associating a unique identifier (UID) to each one. Using services of this naming system, coauthors can identify, contact, and send information to other coauthors regardless of whether they are working at the same time or not.

Using this naming system, an author "Juan" whose login is "juan", who works on the "sun.cs.cinvestav.mx" site, and who stores his documents on the "moon.cs.cinvestav.mx" site, is defined as:

Author	Juan
StorageUrl	http://moon.cs.cinvestav.mx:1968
WorkingUrl	http://sun.cs.cinvestav.mx:2001
LoginName	juan

The naming function identifies authors, document environments, and document resources. It is based on CGI script syntax as shown in Figure 3.2. The first part of name corresponds to the HTTP server identification (protocol, Internet site id, and port number) on which the document is stored. The next part is the PIÑAS script key that indicates to the server that accesses via this URL must be treated by a dedicated CGI script. The last part is the author logical name.

The architecture of our application, as shown in Figure 3.3, is composed by: a) the AllianceWeb application kernel, b) a Web communication infrastructure, and c) the awareness system.

---

<sup>2</sup><http://www.w3.org/Addressing/>

<sup>3</sup><http://www.w3.org/CGI/>

<sup>4</sup>[www.w3.org/XHTML](http://www.w3.org/XHTML)

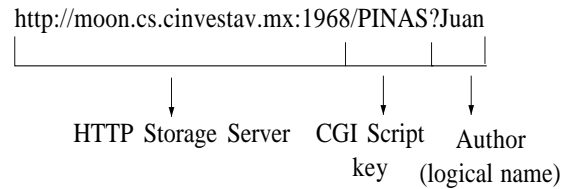


Figure 3.2: Author naming schema

a) **The AllianceWeb Application Kernel.** Because of the unreliability of the Internet support and services, the AllianceWeb editing **functions** are independent of the communication infrastructure (see Figure 3.3). Thus, AllianceWeb is tolerant to network failures, i.e. users can continue to produce in a temporarily disconnected environment. When the connection with the remote server is (re)established, new productions can be obtained and locally validated.

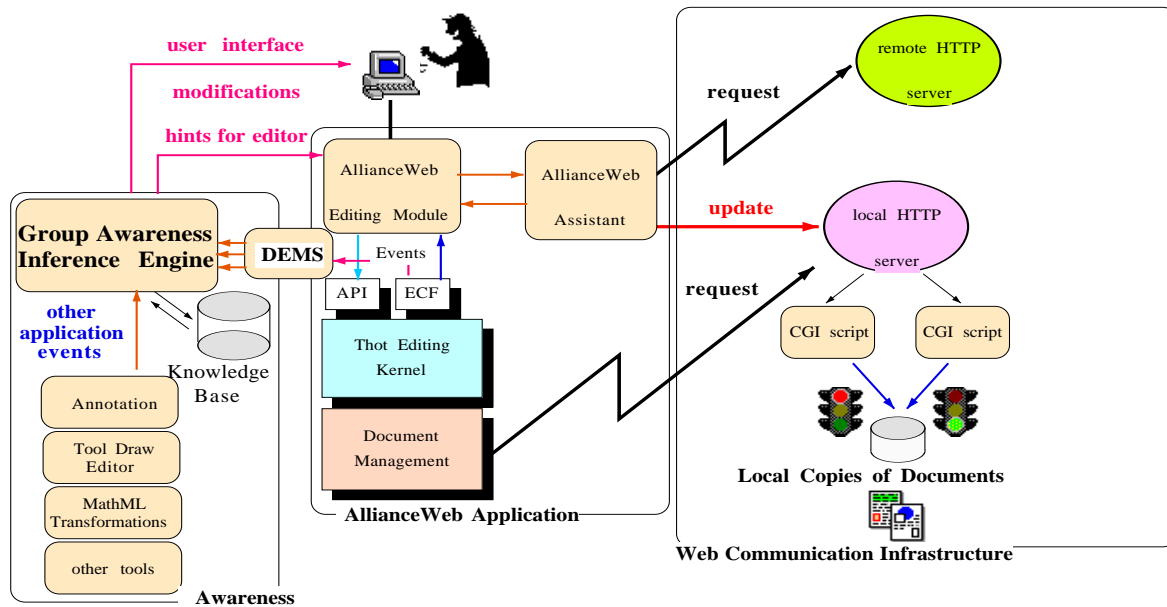


Figure 3.3: Architecture of AllianceWeb

Thus, each running instance of the AllianceWeb application is composed of two separate active parts (processes): the browser and the assistant.

The **browser-editor** component allows document management and authoring. After an editing session, the document is saved. During remote or local access to a document, a collaborator makes requests like to get new role, to update modifications, and to obtain the last version of the shared document. These requests are treated by the AllianceWeb assistant.

In order to facilitate the users with editing functions, the **AllianceWeb editing module** is built on the top of the **Thot Editing Kernel (TEK)** [59]. This Kernel uses a dedicated **document management** module that offers specific functions: naming, document and resource accessing, resource sharing and updating. These dedicated functions are grouped according to handled entities (application, interface, structure tree, and different types of view).

TEK provides highly structured editing functions to handle XHTML documents in a powerful and fine grained way. These functions are accessible using an **Application Programming Interface (API)**.

Another component of TEK is the **External Call Facility (ECF)**. ECF recognizes coauthors' performed actions on the shared document.

These actions are captured in the form of events by a dedicated **Distributed Event Management Service (DEMS)** [12]. The developed DEMS service is responsible for delivering events to event consumer applications, e.g. an awareness system.

**b) The Web Communication Infrastructure.** The AllianceWeb **assistant** makes distinctions between the user working site(s) and the document storage site(s). This assistant is in charge of the communication tasks that are performed in an asynchronous mode.

Documents are replicated on HTTP server sites to enhance their availability, providing support for Web cooperative authoring in disconnected modes: a temporal storage site is then created in the current working site.

In case of communication failure, the assistant repeatedly tries to (re)establish the communication between remote storage servers and local clients.

Coauthors access the shared documents from their working sites. Each shared document is a set of files which contains the document fragments and some metadata: author roles for each fragment, order of all fragments, current state of each fragment, etc. This document information along with the authors roles, is replicated on each of the HTTP server allowing coauthors to work independently on their respective working sites. To ensure the document global consistency, local replicas of shared documents have to be updated each time some remote authors performed and validated their modifications.

**c) The Awareness System.** The AllianceWeb awareness system notifies some interesting facts about users, shared document, and the cooperative environment. The awareness functions are provided by means of an adaptive **Group Awareness Inference Engine (GAIE)**. The GAIE inference engine catches the performed user's actions (in the form of events) as input data, analyzes them using a base of rules (RB), and executes actions providing users with some dynamic functions to adapt their cooperative environment.

The inference rules are written in the first order predicate logic (without functions symbols) [17]. All events are known and are defined in the event dictionary, some facts are previously defined: the coauthor's roles on each fragment of the document, the working sites, and the storage sites. Syntactically, each rule is composed of the premise part and the action part (see Appendix A):

```

rule      ::= Startrule id_rule
           If premise_rule*
           Then
             action_rule*
           Endrule
id_rule   ::= "Sequence of Characters"
premise_rule ::= condition [/* comments */]
condition ::= function_id (argument) relational_operator value
argument  ::= identifier
relational_operator ::= = | < > | <>
value     ::= identifier | constant | expression
expression ::= expression op_arit expression |
            numeric | Num_function_id (argument)
op_arit   ::= + | - | * | /
action_rule ::= action_id ( argument )
            affectation_op value
affectation_op ::= <- | <==

```

Each rule is checked to be free from lexical, syntactical, and semantical errors. During the lexical analysis, it is verified that the events considered in the rule premise, are defined in the event dictionary. An error is pointed out when a fact in the rule differs from those included in the base of facts.

When a user starts the cooperative work session, the GAIE inference engine is installed to inform him/her the activities of other users and vice versa. The inference engine continues functioning until the user terminates his/her work session.

There are other applications that generate events, like **annotation** (tools to cooperatively annotate/comment a shared document with multi-media), **tool draw editor** (tool to draw SVG graphics), and **MathML<sup>5</sup>transformations** (an assistant that allows users to get a MathML expression from the given one reusing its components). These events are captured by the DEMS service and are delivered to the inference engine which must be considered as an event consumer.

The remaining of this chapter explains the principles of the GAIE inference engine, its goals, and the means by which it achieves them.

---

<sup>5</sup>[www.w3.org/Math](http://www.w3.org/Math)

### 3.3 Principles of the Group Awareness Inference Engine

The implemented group awareness functionalities determine handled shared entities (shared document, its fragment and resources<sup>6</sup>, user's definition) to follow actions applied on them in order to infer, to synthesize and to present information about authoring actions. The authoring actions are facts stored in the knowledge base of the GAIE inference engine. From these facts, the inference engine deduces individual or common properties related to collaborators and/or to the shared resources.

The design and implementation of the inference engine is based on the following principles:

1. **Information catching.** The retrieval of information in the form of events for the deduction of new facts that can be transmitted to collaborators. For example, during a cooperative authoring process, the document opening and closing constitute two interesting points to qualify and/or to characterize each user production by means of which a user is deduced as a possible expert on a particular area, e.g. producing Math expression. For this purpose, the inference engine takes into account the following facts: 1) the role of a coauthor, 2) the nature of the produced section, 3) the performed actions, 4) the number of produced well-formed formulas, 5) the computed complexity of each one, and 6) the status of the database to save the author's definition as we see in the "Defining expertise on writing formulas" rule.

```

Startrule "Defining expertise on writing formula"
  If author(fragment_1) = x /* 'x' is author of fragment_1 */
    role(x) = "Writer"      /* 'x' has the writer role */
    nature(fragment_1) = "formula" /* a formula is produced */
    update(fragment_1) = "true" /* formula is saved */
    computed_complexity(fragment_1) > 16
      /* complexity of each formula is calculated */
    summary_wff(formula) > 5      /* formulas are counted */
    status_open(authors_definition_data_base) = "true"
      /* author's definition file is open */

  Then
    author_definition(x) <= "expert_MathML"
  /*'x' is defined as an expert in writing mathematical formula */

Endrule

```

The rule shows that when a coauthor produces five well-formed formula, and the complexity of each formula is greater than 16, he/she is deduced as an expert in

---

<sup>6</sup>The resources of a document can be images, tables, formulas, etc. which can be referenced by another document.

producing formulas. The facts about how a formula is determined to be well-formed and how its complexity is calculated, are respectively treated in Chapters 4 and 5.

2. **The deduction of new facts.** From the caught facts (opening and closing session, data consultation, or modification, deleting, inserting, clicking, etc.), new more elaborated facts are deduced. These new facts may eventually be processed by rules. The inferred facts transmitted to coauthors allowing them to adapt their environment.

For example, a user spending more time in his/her production is deduced that he/she is in trouble or he/she is inactive during the cooperative session.

3. **The proposition of actions.** The GAIE inference engine proposes some tools and communication services in order to enhance cooperative production. For example, when a user is deduced to be in trouble, he/she is notified of the existence of an expert and the former is proposed to communicate with the latter.

Depending upon the role, nature of the fragment (document sharing and editing grain), actions performed by coauthor on the fragment, the GAIE inference engine deduces new facts and notifies them to the concerned users.

### 3.4 Goals of the Group Awareness Inference Engine

The GAIE inference engine aims to make the cooperative authoring environment more flexible and adaptive enabling each collaborator to export his/her contributions, and to import other user contributions to integrate them into the shared production in a comfortable way. More, he/she takes benefits from the automatic updating of the cooperation interface. User is discharged of all cooperation support management duties and his/her actions are then directed only to the cooperation goals.

The inference engine reevaluates and updates the set of manipulated objects (coauthors, private and/or shared objects). This periodic reevaluation is performed in parallel with actions and processing applied by users (e.g. "Jean" in Figure 3.4). This reevaluation is completed by statistic measures of the manipulated/handled object space and by the analysis of concurrent interactions (concerned users, manipulated objects and action frequency).

Suppose that "Jean" mainly acts as a consultant (reviewer or annotator) on the document whereas the author "Joelle" modifies it. Whenever "Jean" opens the document, the information is captured, filtered and presented to "Joelle" who can communicate with the consulting user. Therefore, the GAIE inference engine intends to improve coordination between two coauthors depending upon their communication interest and working states, e.g. a user is referring a figure while it is being modified currently by other partner.

The goals of the GAIE inference engine are:

1. **To enhance group awareness.** In the context of cooperative authoring, the group awareness refers to maintain regularly updated information about the performed actions of each coauthor on the shared object(s). Thus, the GAIE inference engine



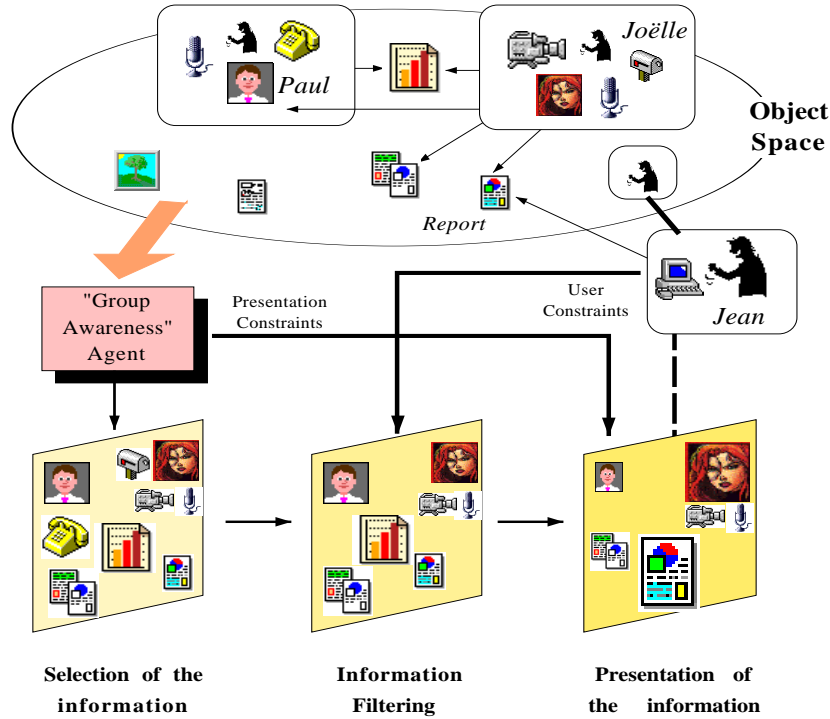


Figure 3.4: Information selection, filtering and presentation

targets to provide information to coauthors about activities of their colleagues. Additionally, the inference engine notifies coauthors the communication interest of users, the produced objects, and coauthors' inactiveness during the cooperative session.

2. **To assist coauthors.** The GAIE inference engine is designed to assist coauthors. It provides the information about tools for handling images, tables, and mathematical expressions. In addition, users can transform MathML [8] formulas from one pattern to another one (Chapter 4).
3. **To improve the coordination.** The GAIE functions target to determine the expertise level of a user in some particular fields like producing mathematical formulas. Previous and current production of a coauthor is analyzed to determine his/her production capabilities (Chapter 5).

## 3.5 Enhancing Group Awareness

"Awareness is an understanding of the activities of others, which provides a context of your own activity" [55]. The activities performed by a person affect the doings of his/her closed collaborators. It is important for all members to have information about activities and work results of their colleagues, which are related to their production. For example,

notice board, activity list, evolutionary report, among others, are formal ways to inform and to initiate members to produce.

Coauthors may need the following information when they produce within a cooperative authoring environment:

- *When does a user want to communicate with his/her colleague(s) under a specific context?*
- *When does a user modify a fragment that is referred by his/her colleague(s)?*
- *When does a coauthor join and leave the cooperative work session?*
- *What does a user produce? Or is he/she inactive within a cooperative session?*
- *Does a user want to maintain privacy regarding his/her performed actions on the cooperative production?*
- *What is the class of a shared document recently opened by a user?*

In order to improve the individual contribution, it is essential to control the quantity of available information and to provide particular facts to each group member, according to his/her working environment. Being aware about activities of others makes one's work more fluent and efficient within a group organization. Moreover, having up-to-the-minute information about what is happening in the cooperative workspace reduces the risk of incoherent production [44].

The desire of individuals to have the knowledge about activities of others (group awareness) has gained importance, such that all groupware applications must include this kind of functionalities. Awareness is not supposed to merely notify the presence/absence of group members, additionally, it should provide the facts about the shared production like what has been produced by whom, in how much time.

In order to enhance group awareness in the cooperative authoring environment, the following objectives are established:

- Establishing a context-based communication between two coauthors by deducing their "work focus".
- Maintaining the consistency of the cooperative production by determining the "work proximity" of interrelated objects.
- Importing and exporting the preferences of the users' environment. This is performed by notifying the coauthors the interesting facts: opening-closing a shared document, starting-leaving session, the privacy of coauthors, and classification of a shared document.

The inference engine optimizes communication between two users when they manifest specific interest to establish a communication centered on a particular context as we see in the following scenario:

*Jorge reviews a document section while Carmin, with the writing role, edits the same section. Jorge wants to enquire/suggest a modification in this fragment. He desires to highlight his focus of discussion on Carmin's display and to establish a synchronous communication with her.*

This scenario shows the main need of coauthors to establish a synchronous communication centered under a particular focus within the shared production. We describe how the GAIE inference engine determines the "work focus" on which users have the possibility to discuss/communicate.

### 3.5.1 Work Focus and Perception

Collaborators can review a document section accessed by other users. Reviewing the document, the reviewer can suggest some modifications, and inevitably they may need to establish a synchronous communication with the producers, to precise and comment observations. Considering this need, we integrate a context-based communication service in the AllianceWeb cooperative authoring application.

The **context-based communication** is defined as: "*A quasi synchronous exchange of messages between two coauthors over a produced object (a formula, a table, a figure, a text) within the shared document*". The object on which the coauthors center their discussion is called as "**work focus**". During this communication, the common document can be displayed in the working environment of two collaborators, and the GAIE inference engine takes advantage of the unique identifier associated to each component of the shared document to transmit the "work focus" of their common and individual working interests.

To support a context centered communication between two collaborators, the inference engine takes into account the following conditions:

1. The two coauthors joint the cooperative session.
2. The two coauthors open a synchronous communication.
3. One of these coauthors opens the inter-communication menu-box to send the selected object (work focus) to the other coauthor.
4. The other coauthor opens the inter-communication menu-box to get the selected document part (localizable using its unique identifier, id).

The reader interested to transmit his/her observations, takes the initiative to establish the communication, and selects the section(s) of the document on which he/she wants to focalize (see Figure 3.5). Selecting a document part, the user highlights one or more textual, mathematical or graphical elements. In case of partial selection, the engine obtains the positions of the first and the last selected parts. The selection is usually defined by

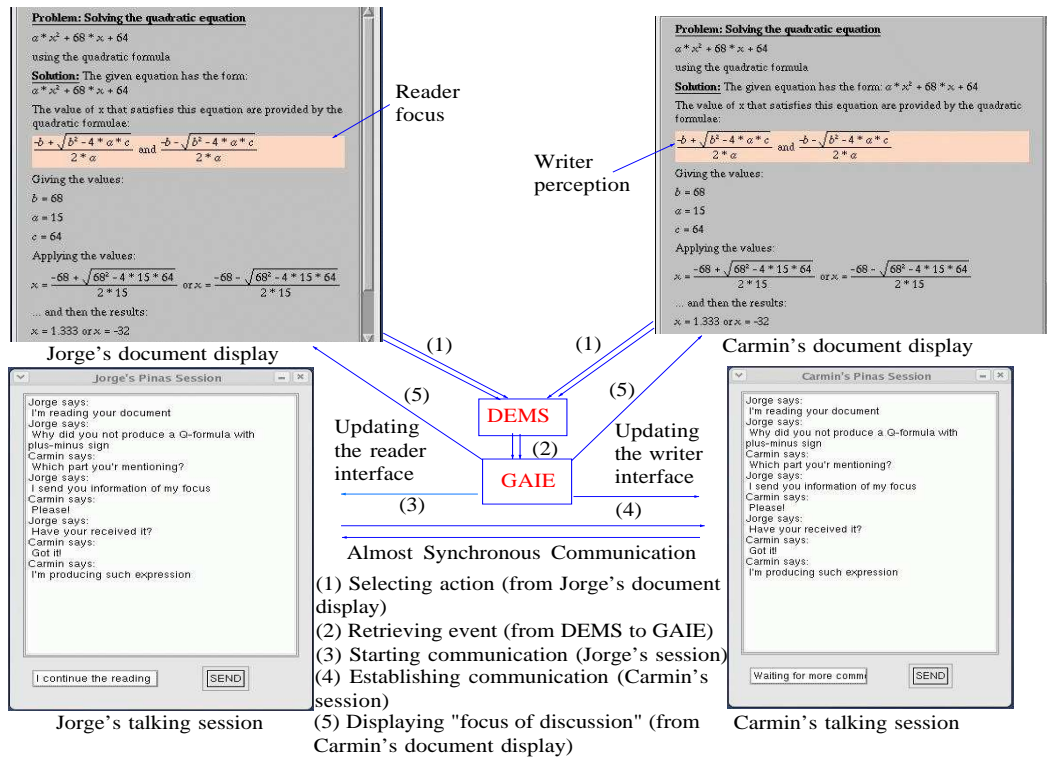


Figure 3.5: The "work focus" and "perception" during synchronous communication

holding the right mouse button and dragging up or down. The same action can also be done by continuously pressing the "Escape" key until the desired element is selected.

As the inference engine receives authoring events from the DEMS service, it obtains the unique identifier of the selected part (figure, formula, table, etc.) and then it may transmit this information to other environments where the communication takes place.

As the other coauthor can open and then view the same document on his/her display, he/she is notified of the potential communication, having the opportunity to accept the synchronous communication. After the communication is established, the selection will be displayed in both working environments. More generally, users can continue their synchronous communication complemented by a dynamic "work focus" having a consistent perception of the shared document. The following rule is applied:

```
Startrule "Work focus communication"
```

```
  If author(fragment_1) = x
    author(fragment_1) = y
    role(x) = "Writer"
    role(y) = "Reader" /* y can read fragment_1 */
    sync_comm(x) = "true" /* communication is enabled */
    sync_comm(y) = "true"
    action(y) = "select_element" /* action of y is selection */
```

```
Then
```

```

display(y) <- send(id)
/* Unique identifier of the selected section is recovered and transmitted */
announce(y) <- "Information is successfully sent"

```

Endrule

The semantic of the conditions of this rule is as follows (see Appendix C):

- $\text{author}(\text{fragment}_1) = x$ , defines the set of  $x$  of coauthors and their roles on the document fragment,
- $\text{role}(x) = \text{"Writer"}$ , defines the set of  $x$  of coauthors having the role 'role\_value',
- $\text{sync\_comm}(x) = \text{"true"}$ , defines the coauthor  $x$  has the synchronous communication available, and
- $\text{action}(x) = \text{"select\_element"}$ , defines the coauthor  $x$  is performing writing action "action\_name".

The semantic of the actions of this rule is (see Appendix D):

- $\text{display}(x) <- \text{send}(\text{id})$ , displays the "work focus" in the coauthor's environment, and
- $\text{announce}(x) <- \text{"Information is successfully sent"}$ , notifies a user of different activities of the coauthors in the cooperative authoring process. The user who is intended for notification becomes the argument of the "announce" action and a string of characters becomes the notification message sent to him/her.

The "*Work focus communication*" rule shows that a coauthor "x" produces a fragment and another user "y" reviews it. The synchronous communication between two users is enabled and the reader selects a fragment as a "work focus" to take it as a matter of discussion. The inference engine recovers the unique identifier (id) associated to the selected element and sends this information to the producer's environment

Following we describe how the "work focus communication" proceeds:

Suppose, Jorge is reviewing a document mainly composed of formulas and he wants to recommend the producer "Carmin" to write an alternate expression. She asks him to send the new focus of discussion. Jorge, who has started a synchronous communication session with Carmin, selects the new "work focus". The GAIE inference engine recovers the unique identifier of the selected section and displays it in the Carmin's environment, as a result of the "*Work focus communication*" rule applied. The process of "work focus" selection and perception is as follows (see Figure 3.5):

1. Selecting action. The reader selects the section of the shared document on which he/she wants to focus (Jorge's document display).

2. Retrieving event. The GAIE inference engine retrieves events from the DEMS service (from DEMS to GAIE).
3. Starting communication. The reader starts communication with the producer (Jorge's session).
4. Establishing communication. A quasi synchronous communication is established between the reader and the producer (Carmin's session).
5. Determining the "work focus" under discussion. The inference engine displays the fragment selected ("work focus") by the reader on the producer display (Carmin's document display).

When a user performs modifications on a document section/fragment that affect the current contextual "work focus", the inference engine updates it in all concerned display/environments. For instance, changes in a figure entail modifications in its description and hence, the concerned collaborators are notified about this fact. When coauthors do not agree with these modifications, they can initiate a synchronous communication with the producer. Then the following rule is applied:

```
Startrule "Perception of a coauthor"
  If author(fragment_1) = x
    session(x) = "true"
    sync_comm(x) = "true"
  Then
    announce(x) <- "Information received for the work focus"
    display(x) <- perceive(selected_element)
    /* 'x' perceives the work focus*/
```

Endrule

When a coauthor is active in a work session and the synchronous communication service is enabled, he/she is notified about the receipt of "work focus" sent by another author with whom the communication has been established. The "work focus" is displayed in the concerned environment, as a result of the "*Perception of a coauthor*" rule.

The individual presence notification is resolved by the Internet instant messaging services (e.g. MSN<sup>7</sup> messenger) [21]. The user of this service has a list of his/her colleagues with whom he/she wants to talk occasionally. When a user starts the session, the notification is sent to all on-line users, as shown in Figure 3.6. Each user can see the current status (on-line/off-line) of all colleagues. The present users may be active, away or busy, depending upon their chatting states. Users can also change their status to inform their colleagues. For example, when they have to leave the office, they can announce: "not in office", "out for lunch", etc.

---

<sup>7</sup><http://www.msn.com>

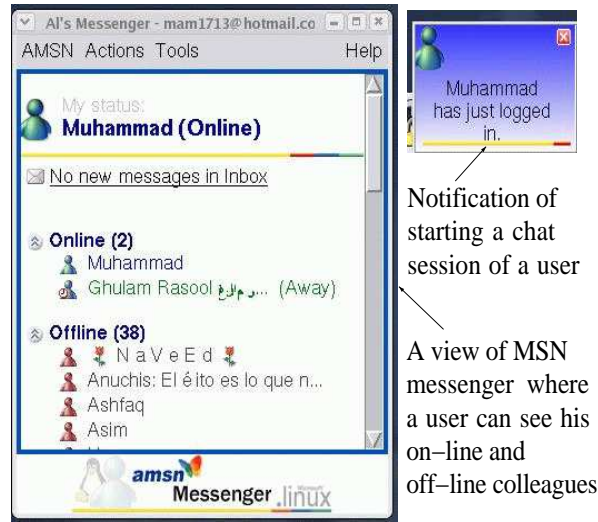


Figure 3.6: MSN messenger notification for starting session of a user

In our platform we have integrated this instant messaging service, such that when a user opens a collaborative session, this service is automatically launched. The coauthor can communicate with his/her colleagues under a "work focus" of discussion.

**Dynamic Work Focus.** Users can dynamically select different sections of the shared document. At one time they can highlight a formula, or a table, or any part of a figure, and at the next moment they can select another one. They can also select at the same time mathematical, graphical, and textual elements, since our awareness system only transmits the unique identifier of this selection. The GAIE inference engine recovers this identifier to display the selected section.

When an author modifies and updates the shared document, the other coauthors receive the notification of this fact. Once coauthors reload the modified document, they get the updated version. This scenario can be seen in Figure 3.7.

Suppose, *Mark* reviews a complex table produced by *Sara*. The reviewer has a confusion about one column and he wants to discuss this point with the producer, opening a communication session. *Sara* requests him to select the "work focus" and to send it (*Sara's* communication session). *Mark* selects the column (*Mark's* document display) and sends it to *Sara*. Using authoring events and the unique identifiers of the selected element(s), the same column is highlighted in the producer display/environment (*Sara's* document display in Figure 3.7).

But, *Sara* has performed some modifications on the table, these changes are not available to *Mark* until she updates the document. The modifications can be viewed in the *Sara's* document display (Last Column). Once the producer updates the fragment, readers can recover it and precise the "work focus".

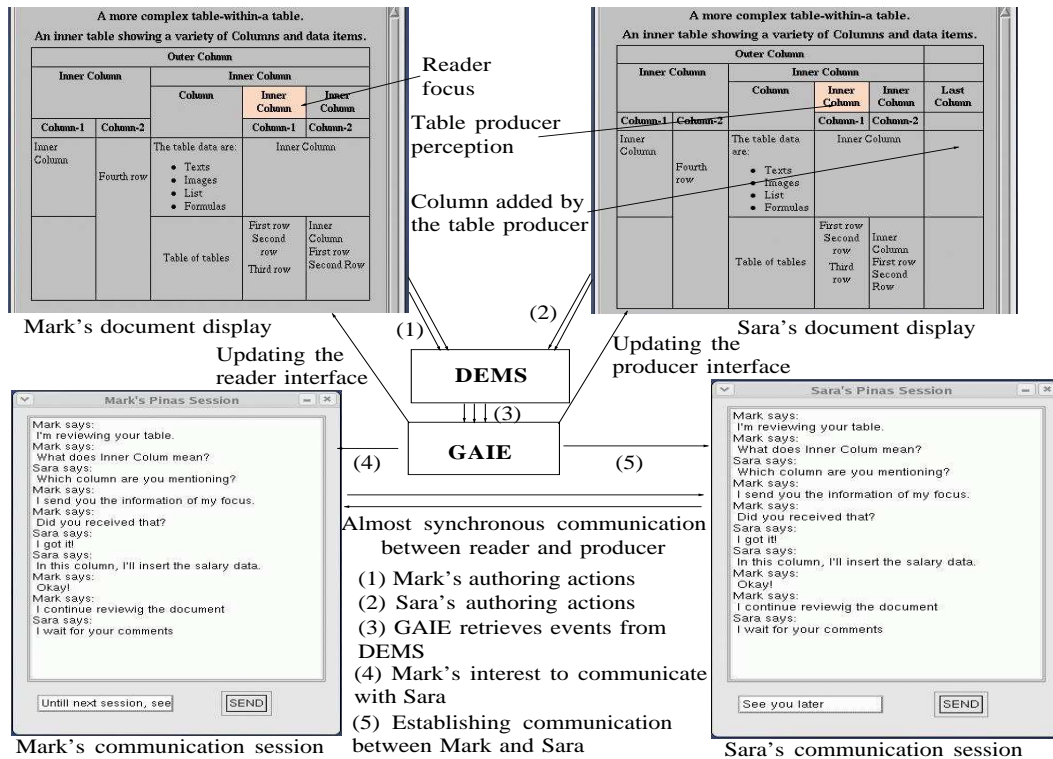


Figure 3.7: The "work focus" and "perception" on a formula based document

**Related Work.** The asynchronous communication facilities (e.g. e-mail) has been integrated into BSCW [3][4][1]. The BSCW users can send/receive many short/long messages when they want to discuss some points within the shared document.

In contrast to BSCW, in our approach, the GAIE inference engine determines the communication interest of coauthors and provides context-based communication. This communication allows coauthors to immediately center the discussion and settle a matter within the shared production.

### 3.5.2 Work Proximity

In cooperative authoring, a shared document can have interrelated sections, for instance, a table, a figure, an image, or a formula, and its respective legend. In addition, these sections can be produced by different coauthors. When an interrelated section (a figure) is modified by its producer, the notification functions integrated in our awareness system must notify this fact to the concerned coauthors (the producer of the figure legend).

In general, "when at least two objects are perceived as belonging to a unit, we say that these objects are near/close". We denominate this property as "work proximity".

For instance, a coauthor takes the responsibility to produce an elaborated figure as shown in Figure 3.8 (a) while his/her colleague produces the associated legend Figure 3.8 (b). A modification of the figure may entail a change in its description. Thus, in order to



maintain consistency of the shared document, the GAIE inference engine determines who are the concerning authors: the drawer of the figure and the author of its explanation.

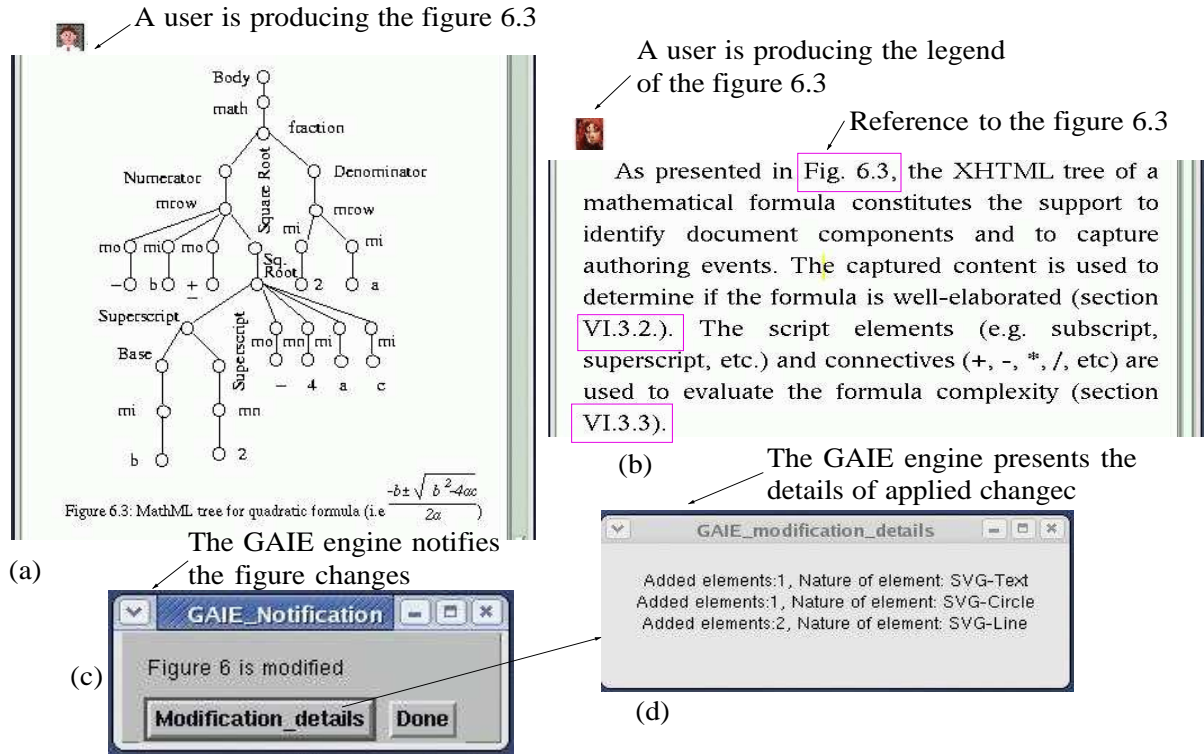


Figure 3.8: The "work proximity" notification

The GAIE inference engine determines the "work proximity" among related fragments by means of the following rule:

```

Startrule "Related fragments"
  If author(fragment_1) = x
    role(x) = "Writer"
    nature(fragment_1) = "image" /* fragment_1 is an image */
    update(fragment_1) = "true" /* fragment_1 is saved */
    author(fragment_2) = y
    role(y) = "Writer"
    nature(fragment_2) = "text" /* fragment_2 is a text */
    reference(fragment_2) = fragment_1
    /* fragment_2 refers fragment_1 */
  Then
    announce(y) <- "referred object is modified"
Endrule
    
```

Each time a referred section is modified, this fact is notified to coauthors that make reference to this section, by using the "Related fragments" rule.

When a user opens/closes a shared document, the GAIE inference engine summarizes its contents. When the number of components of a specific object (a SVG graphic, a Math expression, and XHTML structures) are decreased (respectively increased), the engine notifies this fact as shown in Figure 3.8 (c). The interested coauthor(s) can also obtain the details of modifications as shown in Figure 3.8 (d).

The modifications of the shared production can be represented as:

$$\Delta E = |S_{v,1}| - |S_{v,2}|$$

where,  $\Delta E$  = number of modified elements,

$|S_{v,1}|$  = Cardinality of the set of elements in the previous fragment version,

$|S_{v,2}|$  = Cardinality of the set of elements in the current fragment version.

The coauthors must take into account their respective "work focus", since it is quite possible that the work performed by one of them affects the production of the others. The coauthors must be aware of their production on the interrelated document sections.

**Related Work.** One of the most important research related to our approach is **Equi-Text** [60][61]. It informs the performed actions on the shared objects in chronological order by using the function "records". The performed actions are: - **insert** to add a new paragraph after the current one; - **alter** to change the contents of a paragraph; - **exclude** to confirm the exclusion of a paragraph for edition appearing in a fading color; etc.

In our approach, the GAIE inference engine provides the details of all changes performed on the shared document by a particular user: addition and deletion, precise characteristics of accessed document elements during a session, etc.

The diversity of the provided information by the inference engine is very useful for coauthors in order to maintain consistency in the cooperative production.

### 3.5.3 Opening-Closing Document and Starting-Leaving Session

Normally when coauthors produce within a local area network, they may have the interest to be informed when a particular coauthor starts a cooperative session or opens a specified shared document. Additionally, when a user is present in a cooperative session, the details of his/her production is highly important for control and management purposes.

The notification of starting-leaving the cooperative session and opening-closing a shared document are important for users producing in a broad area (Wide Area Network, WAN). The remote users can be notified by sending information via e-mails. The provided information may include: the name of coauthor, the document accessed, date and time to start and terminate the cooperative session, as well as the working site. By means of this kind of notification, coauthors can timely discuss ideas and can ask for guidance from others.

When a coauthor starts the cooperative session (respectively closing) and opens a shared document (respectively close), some events are produced (see Figure 3.9). The following rules use these events:

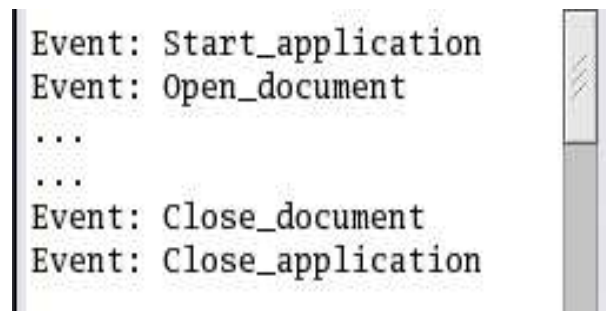


Figure 3.9: User authoring actions to start and terminate the cooperative session

```

Startrule "A reviser opens a shared document"
  If author(fragment_1) = x
    role(x) = "Writer"
    session_on(x) = "true"
    author(fragment_1) = y
    role(y) = "annotator"
    annotation(fragment_1) = "true"
    /* the author 'y' can read and annotate the fragment_1 */

    action (y) = "open_document"
  Then
  announce(x) <- "annotator opens document"
    /* author 'x' is informed that author 'y' starts the session */

Endrule

Startrule "Release of a shared document"
  If author(fragment_1) = x
    role(x) = "Writer"
    session_on(x) = "true"
    annotation(fragment_1) = "true"
    role(y) = "annotator"
    action(y) = "close_document"
  Then
  announce(x) <- "annotator closes shared document"
  /*coauthor 'x' is informed that coauthor 'y' closes the document */

Endrule

```

The above rules inform coauthors when a reviewer opens/closes the shared document.

The names of the coauthor, the document name, the working site, and the storage site, are informed to other coauthors as shown in Figure 3.10. If a coauthor wants, he/she may communicate with the coauthor who has recently logged in.

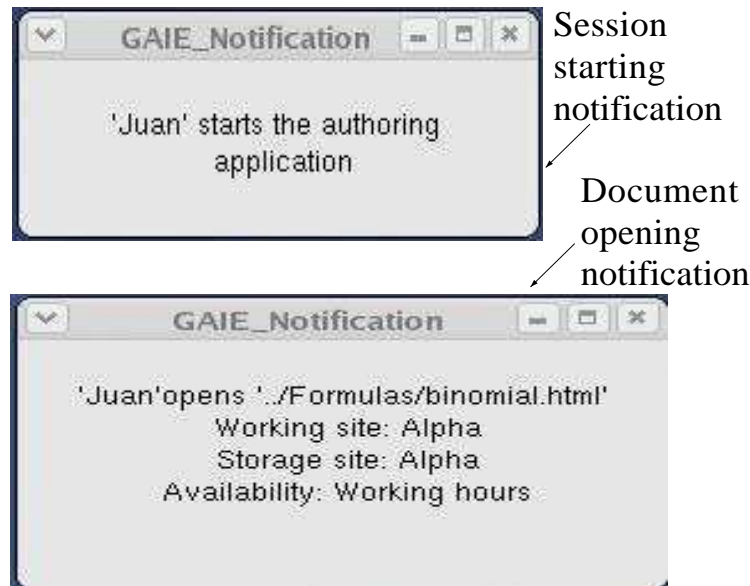


Figure 3.10: Session start and document open notification

### 3.5.4 Notifying Inactiveness within a Cooperative Session

Being present in the cooperative space, a user may not produce, i.e. he/she becomes inactive in the session. It is quite possible that the user moves to some personal activities performed outside of the cooperative session.

The activity of a coauthor is checked by observing his/her production during a certain time. The increase or decrease in the size of the fragment is used for this purpose.

In order to notify the inactiveness of a user, the engine first determines the idle time and the production of a user by means of the following rules:

```

Startrule "Recording idle time and production"
  If author(fragment_1) = x
    role(x) = "Writer"
    start_session(x) = "true"
  Then
    initial_time(x) <- current_time(x)
                                /* the current time is recorded */
    session_on(x) <- "true"
                                /* to record that session is on */
    meta_data(fragment_1) <- summary(document)
                                /* the contents of the document are registered */

Endrule

```

The GAIE inference engine registers the current time and the summary of the shared document, when a user starts a cooperative session by means of the *"Recording idle time*

and production" rule.

The meta-data of the shared document contains the number of elements composing it, percentage of elements of each MathML, SVG, and XHTML schema, *well-formed* expressions, tables, etc.

The following rule is applied to calculate the idle time and production of a user in terms of added or modified elements:

```
Startrule "Calculating idle time"
  If author(fragment_1) = x
    role(x) = "Writer"
    session_on(x) = "true"
  Then
    idle_time(x) <- current_time(x) - initial_time(x)
                                /* the idle time is calculated */

    production(fragment_1) <- statistic(fragment_1)
                                /* the contents of the document are registered */

Endrule
```

Once the idle time is calculated, the inactiveness of a coauthor can be determined. The following rule is applied:

```
Startrule "Inactiveness"
  If author(fragment_1) = x
    role(x) = "Writer"
    session_on(x) = "true"
    idle_time(x) > 10          /* time in minutes */
    author(fragment_1) = y
    role(y) = 'Manager'
    production(fragment_1) = Null
    communication_interest(y) = x
                                /* the coauthor 'y' wants to communicate with 'x' */

  Then
    announce(y) <- 'production of coauthor 'x' is null'
                                /* the 'x' is inactive */

Endrule
```

The "*Inactiveness*" rule notifies the coauthor 'y' (see Figure 3.11) when a coauthor "x" working for more than 10 minutes, has not produced. By clicking the "Information button", the inactive user data are displayed. An e-mail message can be generated at the recipient's terminal of the concerned user by clicking the "Send Message button". The availability of cooperating users depends on their working routine, office hours, organization steps, etc.



Figure 3.11: Present activity of coauthor

The manager of the cooperative work group warns against inactive users to encourage them to complete their task, and to unlock the assigned fragment. In case of other coauthors provide evidence for wanting to collaborate on the assigned section on which a user is inactive, the manager can decide to unlock this fragment and may assign it to another coauthor.

### 3.5.5 Privacy and Intrusion

Due to the fact that the users focalized on their work, may consider that to be informed each time their colleagues join or leave the cooperative space, is an intrusive information, the awareness notification functions are subjected to coauthor's preference filters. On the other hand, a user may disable the awareness notification functions, thus his/her colleagues cannot be aware about his/her entrance. The GAIE inference engine may ensure privacy and isolation, if it is desired.

An author who disables the notification service will not be aware of the actions/production of other collaborators as a result of the *"Notification service"* rule.

```
Startrule "Notification service"
  If author(fragment_1) = x
    role(x) = "Writer"
    session_on(x) = "true"
    notification(x) = "disable" /* x disables notifications */
    author(fragment_2) = y
    role(y) = "Writer"
    session_on(y) = "true"
    communication_interest(y) = x
  Then
    announce(y)<- ''The notification service is disabled''

Endrule
```

### 3.5.6 Classifying a Shared Document

The classification of a document allows users to have a quick judgement about its content. When a document is classified as mathematical, graphical, or descriptive, it refers

that most of its contents respectively include formulas to prove theorems and corollaries, drawings (images, lines, arrows, colors, etc.), or textual components (sections, paragraphs, lists, etc.). When, the user comes to know the document kind, he/she may plan his/her activities towards the cooperative production.

On our authoring application, the classification of a shared document is related to assigning a category to it, based on its contents. The included contents represent XHTML, MathML, and SVG structures that are respectively used to produce structured texts, Math expressions, and vector graphics. Hence, the developed group awareness functionalities analyzes the shared document, and classify it according the prevailing contents.

To provide useful information about the nature of the elements composing a document, the inference engine uses the following rule:

```
Startrule "Summary of a document"
  If action(x) = "open_document"
  Then
    meta_data(document)<- summary(document)
      /* The elements added or deleted from the document */

    announce(x) <- meta_data(document)
      /* Number of formulas, links, images, paragraphs */
```

Endrule

**Problem: Solving the quadratic equation**  
 $a * x^2 + 68 * x + 64$   
 using the quadratic formula  
**Solution:** The given equation has the form:  $a * x^2 + 68 * x + 64$   
 The value of x that satisfies this equation are provided by the quadratic formulae:  
 $\frac{-b + \sqrt{b^2 - 4 * a * c}}{2 * a}$  and  $\frac{-b - \sqrt{b^2 - 4 * a * c}}{2 * a}$   
 Giving the values:  
 $b = 68$   
 $a = 15$   
 $c = 64$   
 Applying the values:  
 $x = \frac{-68 + \sqrt{68^2 - 4 * 15 * 64}}{2 * 15}$  or  $x = \frac{-68 - \sqrt{68^2 - 4 * 15 * 64}}{2 * 15}$   
 ... and then the results:  
 $x = 1.333$  or  $x = -32$

Information of currently open document  
 [11] formulas [0] SVG elements  
 [0] images [14] paragraphs  
 [0] links [0] tables  
 Percentage of:  
 HTML-elements = [16.41%]  
 MathML-elements = [83.59%]  
 SVG-elements = [0.00%]  
 The document is mathematical type  
 \*\*\*\*\*

The summary of the recently opened document

A document being edited by a coauthor

Figure 3.12: Classification of a document

Figure 3.12 presents an example of the notification giving by the "Summary of document" rule when a user opens the document. The percentage of the different schemas

are ordered and when the difference between the two more greater is at least 20%, the document is classified as this kind of document (like **H**-XHTML, **M**-MathML, and **G**-SVG). In our example, there are 11 formulas in the document representing 84% of the document, and the textual elements only represent 16%. Hence, the document is classified as mathematical: M-MathML kind.

Let us suppose the task assigned to a user is to describe all the formulas included in a document, then each time, he/she opens it and the summary of the elements composing it is displayed, he/she will know the quantity of the remaining task.

We opened several documents in our authoring application. The inference engine gets the composition for each one as presented in the Table 3.1. The provided information is: the total number of included elements (XHTML, MathML, and SVG schemas), percentage of elements of each schema, number of images, tables, and links. The inference engine notifies the document type depending on the percentage of each schema.

For instance, the document "Doc\_2.html" has 94% of the XHTML elements and 6% of MathML elements, hence the document is classified as H-XHTML kind (textual).

The document "Doc\_4.html" has 31% of XHTML elements, 69% of MathML elements, hence the document is M-MathML kind (mathematical).

The document classification can provide a global view of the common work during either active (authoring sessions) or passive (browsing work sessions) phases. Thus, when authors access a shared document, they become aware of the evolution of the common production.

Table 3.1: Summary of Several Documents

Doc. Name (*.html)	Total Elements (XHTML + MathML + SVG)	Paragraphs (XHTML Elements (%))	Formulas (MathML Elements (%))	SVG Figures (SVG Elements (%))	Images	Tables	Links	Type
Doc_1	77	- (12)(16%)	-	1 (65)(84%)	-	-	-	SVG
Doc_2	2990	43 (2820)(94%)	13 (170)(6%)	-	-	2	-	XHTML
Doc_3	201	13 (38)(19%)	11 (152)(76%)	3 (11)(5%)	-	-	-	Math
Doc_4	259	17 (81)(31%)	4 (178)(69%)	-	-	-	-	Math
Doc_5	3627	153 (3627)(100%)	-	-	-	3	288	XHTML



**Related Work: Notifications in WCWAs.** All Web-based Cooperative Writing Applications (WCWAs) use different notification functions to enhance awareness among collaborators, as we describe as follows:

- Basic Support for Cooperative Work (BSCW) [4][3][1] uses "event icons" to notify several activities of the group members (see Figure 3.13). BSCW distinguishes five types of events represented by icons. Users can get related information by clicking these icons:
  - "New" icon displays the new created object;
  - "Spyglass" icon displays recently downloaded document;
  - "Pencil" icon displays the modified object;
  - "Foot" icon displays the moved object from one folder to another;
  - "Hand" icon displays the modified folder.

Notifications on BSCW through icons are limited to only five events. Users are unable to have information about other cooperative activities, i.e. a user cannot have information whether an object is currently being modified, who is working at this instance, etc.

Our developed approach informs coauthors about all actions performed in the shared space: when a user starts a cooperative session, when he/she liberate a document section, when he/she is active or inactive, and so on.



Figure 3.13: A view of BSCW

- EquiText [60][61] uses an "asterisk" icon to notify some observations (comments) made on the shared document (see Figure 3.14). Clicking on this icon, a user can get the list of all observations. In addition, the current version, and previous version of the document are available through buttons. By contrast, the implemented awareness functionalities in AllianceWeb allow users to get the detail of modifications made on the shared objects by a particular coauthor.

#	Conteúdo do Parágrafo	Colaborador	Ação	Obs	Data
1º	Testando apenas!	<a href="#">jocenildes</a>	I	---	24/05/2005-15:17
2º	blabliblo	<a href="#">gabigorda</a>	I	* _	06/05/2005-17:46

Figure 3.14: A view of the EquiText document

In EquiText, users cannot identify who has made changes and what kind of changes are made.

- REal-Time Distributed, Unconstrained Cooperative Editing (REDUCE) [75][76] uses colors to notify collaborators of the editing tasks. For example, red color is used to show active tasks, and green color indicates the idle tasks. The team members can visualize the editing process in order to create a better teamwork environment.

The developed awareness functions in AllianceWeb notify the evolution of the cooperative production and the actual state of the shared object, whether a coauthor is modifying it, reading or annotating it.

## 3.6 Conclusions

In this chapter, we explained some issues to the solution of group awareness:

1. The coordination of coauthors' activities by means of the *context-based communication*. When a shared object becomes the "work focus" under discussion, our awareness system automatically transmits, distributes, and updates it to the concerned users' display, in order to maintain consistently the communication and to reach an agreement on the cooperative production.

2. The consistence and the coherence of the shared document by determining the "work proximity", when at least two objects (e.g. a figure and its legend) are perceived as belonging to a same unit. Each time, one of these objects is modified, the awareness system notifies this fact to the concerned users.
3. The *notification* of the evolution of the cooperative production and coauthors' productivity.

In the next chapter, we explain the second main goal of the inference engine: to assist coauthors.



# Chapter 4

## Transformation of MathML Formulas

The Group Awareness Inference Engine (GAIE) was designed and implemented to enhance awareness among coauthors. In this chapter, we report the way in which our group awareness system provides assistance for handling MathML formulas: transformation and reuse of formulas.

### 4.1 Introduction

Due to the fact that our authoring application handles the Extensible HyperText Markup Language (HTML/XHTML) [35], it is possible to produce large elaborated documents. To represent mathematical notations, it is used the Mathematical Markup Language, MathML [8]. To produce high quality figures of two-dimension, it is used the Scalable Vector Graphics, SVG schemas [54].

Hence, it is possible to distinguish the writing actions and the nature of the production (e.g. structured text, headings, paragraphs, lists, hypertext links, mathematical statement, figures).

In order to distinguish the writing actions of a coauthor we develop a Distributed Event Management Service (DEMS)[12]. The DEMS filters the information needed by the GAIE inference engine to deduce important facts.

For instance, a user wants to produce an elaborated formula, but he/she spends a lot of time to complete it. The inference engine deduces that he/she has some troubles, then it suggests him/her some suited functions included in the authoring application to improve his production. In addition, the GAIE inference engine provides information about experts in order to contact them (Chapter 5).

On the other hand, it is useful to invoke menu commands directly from the keyboard (keyboard shortcuts or keyboard sequence), but those are not explicitly presented in buttons and type menus, so the GAIE inference engine determines when to notify this kind of information, taking into account the type of writing actions and the current element handled.

In the following, we explain how the GAIE inference engine guides users for handling

formulas.

## 4.2 MathML formulas


Our authoring application uses the Mathematical Markup Language, MathML to write, to represent, and to interpret mathematical statements without ambiguity.

The symbols in MathML are delimited by tags: a number by  $\langle mn \rangle \langle /mn \rangle$ , a variable by  $\langle mi \rangle \langle /mi \rangle$ , an operator by  $\langle mo \rangle \langle /mo \rangle$ , an expression by  $\langle mrow \rangle \langle /mrow \rangle$ , etc.

A MathML formula is delimited by the tags:  $\langle math \rangle \langle /math \rangle$ .

A *well-formed MathML formula* can be produced by completing the following three steps: starting a formula, completing it, and ending it.

### a) Starting a formula

A formula is started by two ways: a) selecting in the "Types Menu", the "Math" option, followed by the "New formula" option; or b) clicking the "MathML icon" (  the first "Math" button in the "Maths box" (see Figure 4.1 (a)).

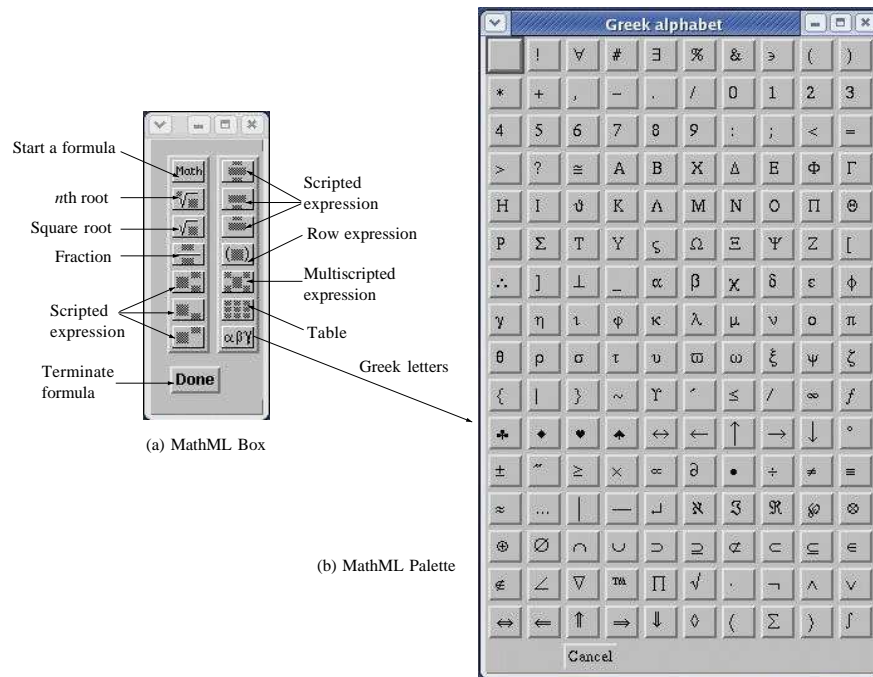


Figure 4.1: MathML box and the greek alphabet palette

### b) Completing a formula

Once a formula is started, it can be completed by: a) using the key board (typing identifiers, numbers, and operators) or b) selecting a pattern (fraction, square root, superscript, etc.) included into the "MathML box". These patterns require one, two or three components (expressions). For example, a square root only requires one component, a  $n$ th

root requires two expressions (a base and an index), and a sub-super-script requires three components (a base, a subscript, and a superscript). In addition, from the MathML box, the greek letters ( $\alpha$ ,  $\beta$ ,  $\lambda$ , etc.) can be inserted, by selecting the "math palette button" (see Figure 4.1 (b)).

### c) Terminating a formula

A formula is terminated by: a) typing the *Enter* key from the keyboard, or b) selecting the "Done" button of the "MathML box".

## 4.2.1 Producing a MathML Formula

To illustrate the above three steps, we explain for instance, the process to produce the quadratic formula 4.1: i) click the "MathML icon", ii) select the "Start button" of the "MathML box", iii) select a fraction pattern, iv) type from the keyboard: " $-b$ ", v) select the last button of the "MathML box" to open the "greek alphabet palette", vi) select the " $\pm$ " sign from this palette, vii) select the square root pattern, from the "MathML box", viii) complete the base expression of the square root from the keyboard, ix) complete the denominator expression of the fraction, x) click the "Done button" which finishes the formula.

$$\frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 * a} \quad (4.1)$$

## 4.2.2 Handling MathML Events

Some specific events are generated and captured during the production of a mathematical expression or when it is selected (Figure 4.2). These events are captured by the DEMS service and sent to the GAIE inference engine, which applies the set of rules to execute specific actions, for instance, the "*Selecting MathML object*" rule.

```
Startrule "Selecting MathML object"
  If author(fragment_1) = x
    nature(fragment_1) = "formula"
    action(x) = "select_element"
  Then
    announce(x) <- evaluate_wff(fragment_1)
    /* The coauthor is informed if a formula is well-formed */
    announce(x) <- list_tools(math_handling)
    /* functions for formulas */

Endrule
```

Thus, GAIE inference engine notifies users the functions integrated into the authoring application: a) validation of a produced expression (well-formed), b) computation of its complexity, c) transformation from one pattern to another, d) rewriting the expression to the prefix and/or postfix forms.

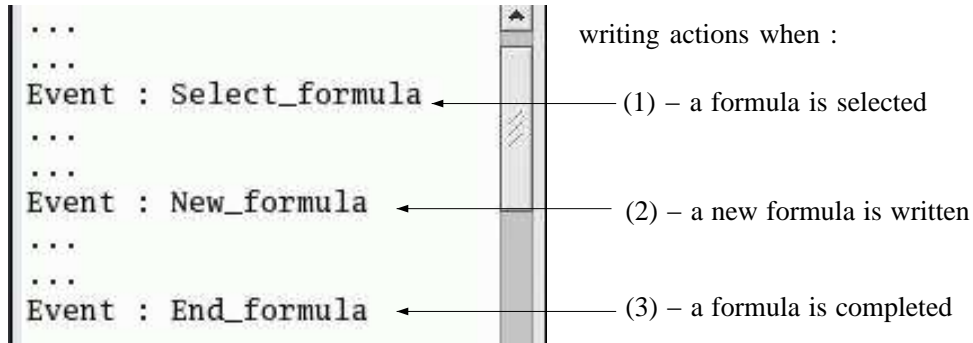


Figure 4.2: Events produced during the production of a formula

### 4.2.3 Well-formed MathML Formulas

A formula is well-formed if all elements composing a MathML pattern are completed and each element, in turn is a well-formed MathML expression too.

For instance, the source view of the formula 4.2 shown in Figure 4.3 is interpreted as follows: a formula is delimited by the `<math>` `</math>` tags; the identifiers *a* and *b* are delimited by `<mi>` `</mi>` tags; the operator/ is delimited by `<mo>` `</mo>` tags.

$$\frac{a}{b} \quad (4.2)$$

```

197 <math xmlns="http://www.w3.org/1998/Math/MathML">
198   <mi>a</mi>
199   <mo>/</mo>
200   <mi>b</mi>
201 </math>

```

Figure 4.3: Source of MathML representation for the formula  $\frac{a}{b}$ 

The "Well-formed formula" rule verifies when a formula is well-formed.

```

Startrule "Well-formed formula"
  If author(fragment_1) = x
    role(x) = "Writer"
    nature(fragment_1) = "formula"
    update(fragment_1) = "true" /* fragment_1 is saved */
  Then
    announce (x) <- evaluate_wff(fragment_1)
    /* Well-formed formulas are announced to 'x' */
Endrule

```

The expression in Figure 4.4 (a) has been produced using the *n*th root pattern, whose



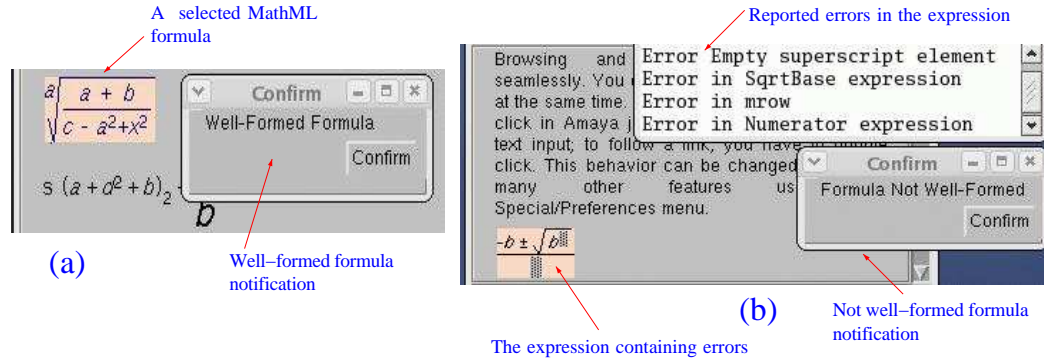


Figure 4.4: Notification: a) well-formed b)not well-formed formulas

components are: the base and the  $n$ th-root. The base is composed by a fraction pattern and the index of the  $n$ th root is a wff. The two expressions of the fraction pattern are also wff's.

Since, all expressions are well-formed, hence the whole formula is well-formed too.

When a formula is not well-formed, the GAIE inference engine notifies this fact: "missing an operator", "bad symbol", "missing ')", "bad operand" (Figure 4.4(b)).

When a formula is started by selecting in the "Types Menu", the "Math" option, followed by the "New formula" option, the produced formula is a MathML sequence like expression 4.3. Once the user changes from the MathML format to another one (text, SVG, etc.), our awareness system considers that the user has finished the expression and it proceeds to analyze the sequence of mathematical characters to determine whether it is a well formed formula.

$$a^2 + +2 * a * b + b^2 \tag{4.3}$$

This expression is declared as not well-formed, because there is not operand between: "+ +"

When a formula or a fragment of it is selected (Figure 4.4(a)), the user may explicitly ask for its evaluation. For instance, in Figure 4.4(b), a not well-formed formula is reported.

Internally, a formula like Figure 4.5 (a) is structured as a tree (Figure 4.5(c)). The source view of this formula is shown in Figure 4.5(b). Each node of this tree is an element container. When a node of the tree is empty (e.g. superscript of the element included in the "square root"), the formula is reported as not well-formed.

The GAIE inference engine analyzes the expression tree from bottom to upward. The identifier "b" is the base of the superscript expression, but the superscript element is missing. Thus, an error is marked in the "Superscript element" entailing the error on "SqrtBase expression" and in the "<mrow>" tag. Finally the error is propagated to the "Numerator element" (see Figure 4.4 (b) and Figure 4.5(c)).

Similarly, the denominator is also reported as "incomplete" since it is empty. Therefore, when an element of the expression contains an error or it is empty, all its father expressions

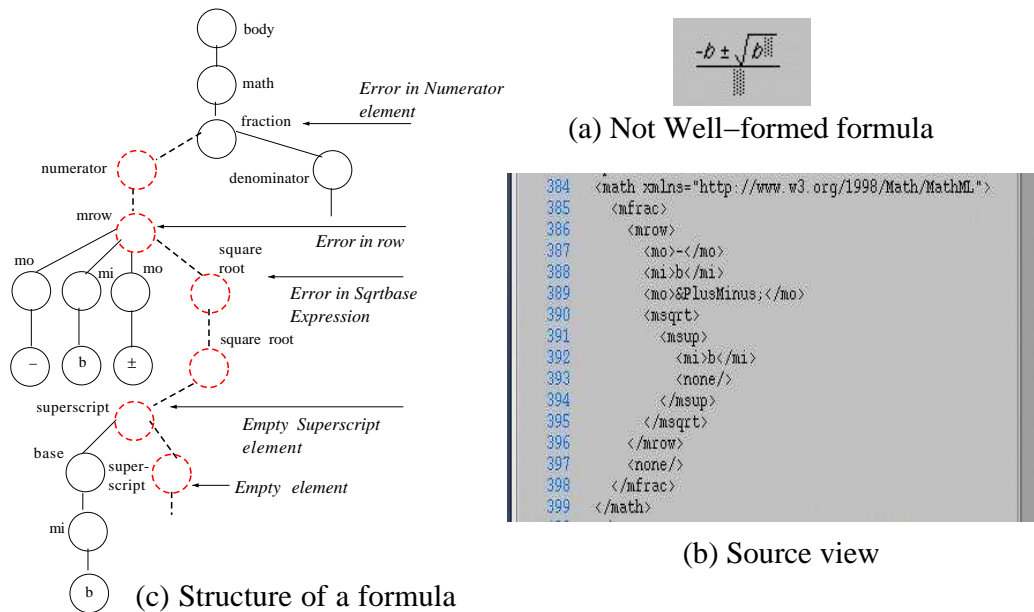


Figure 4.5: An incomplete formula: not-well-formed

are marked as not well-formed.

Once a formula is verified as well-formed, its complexity is calculated by taking into account the involved steps for its production (Chapter 5).

### 4.3 Transformation of a MathML Formula

Due to the structure of a MathML pattern and its representation by a structure tree (see Figure 4.6), it is feasible to *transform* any MathML pattern to another one. The transformation notion we consider here is not related to others well-known transformations existing in several domains such as Mathematics (e.g. expanding  $(a + b)^2$ , factorizing  $a^2 + 2 * a * b + b^2$ , etc.), Logic (e.g. converting a Conjunctive Normal Form (CNF) into a Disjunctive Normal Form (DNF), vice versa, etc.), etc.

The MathML patterns can be characterized by the number of components to be fulfilled. Thus, a transformation is represented by:

$$\text{transformation } ((A, n) (B, m))$$

Where  $(A, n)$  represents the pattern A having n components, that will be transformed to pattern B, having m components.

In the following, we present the studied cases of transforming patterns, according with the number of components to be completed. The pattern in MathML have one, two, or three components (see Figure 4.1 (a)).

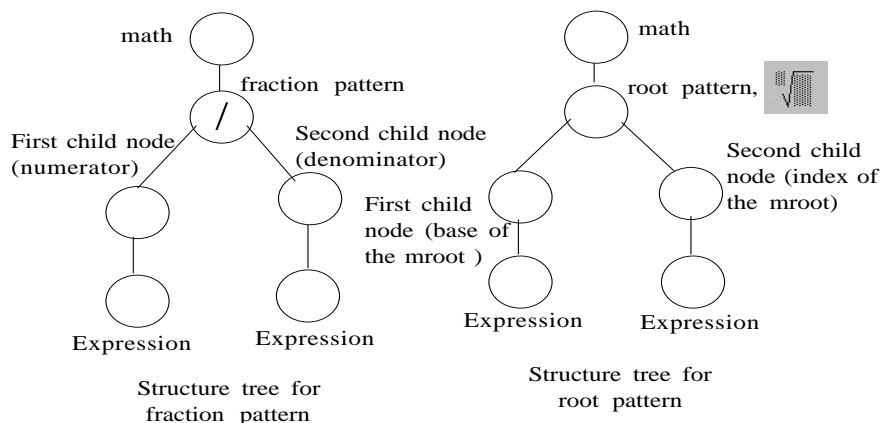




Figure 4.6: MathML pattern structure tree

### 4.3.1 Single Component Pattern

There are only two patterns that have one component: the square root , and the parenthesis symbols . The possible transformation from this kind of patterns are:

#### 1. Transformation ((A,1)(B,1))

In this case, the only possible transformation is to substitute one of the above patterns by the other one. Thus, it is only necessary to move the content of the first pattern (FirstChildNode(A)) to the content of the targeted pattern (FirstChildNode(B)). The "Exchanging patterns" rule is applied:

```
Startrule "Exchanging patterns ((A,1) (B,1))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 1
  Then
    FirstChildNode(B) <- FirstChildNode(A)

Endrule
```

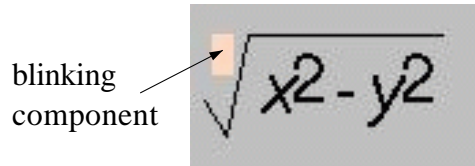
Example: Given the selected expression  $\sqrt{x^2 + 1}$ , its transformation is:  $(x^2 + 1)$

#### 2. Transformation ((A,1)(B,2))

When the target-pattern (B) has two components, the component of the selected expression (FirstChildNode(A)) is assigned to the first node of the target-pattern

(FirstChildNode(B)), and its second node remains vacant (SecondChildNode(B)), so it is maintained "blinking", to indicate that this component must be completed.

As we have two possibilities to fulfill the components of (B), it is necessary to write two rules. In any case, one of the components is kept "blinking", (see the following figure). The user selects one of these possibilities displayed by the rules. When he/she fulfills the empty component, the expression will be considered as well-formed:



```
Startrule "Completing the first component ((A, 1) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 2
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    Blinking (SecondChildNode(B))
    /* Blinking the second empty node */
```

Endrule

```
Startrule "Completing the second component ((A, 1) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 2
  Then
    SecondChildNode(B) <- FirstChildNode(A)
    Blinking (FirstChildNode(B))
    /* the first node is empty */
```

Endrule

Example, given the selected expression:  $\sqrt{x^2 - y^2}$ , and the target-pattern the "under root form", the two possible formulas are:

$$\langle vacantelement \rangle \sqrt{x^2 - y^2}$$

$$x^2 - y^2 \sqrt{\langle vacantelement \rangle}$$

3. (a) Transformation ((A,1)(B,3))

When the target-pattern has three components, there exist three possibilities. Thus, the rules propose three formulas. In each rule, the content of the selected expression is assigned to one content of the target-pattern. The other remaining two children of the target-pattern will be vacant.

```

Startrule "Two remaining vacant components ((A, 1) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 3
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    Blinking (SecondChildNode(B))
                                     /* the second node is blinking */
    Blinking (ThirdChildNode(B))
                                     /* the third node is blinking */

Endrule

Startrule First and third vacant components ((A, 1) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 3
  Then
    SecondChildNode(B) <- FirstChildNode(A)
    Blinking (FirstChildNode(B))
                                     /* the first node is blinking */
    Blinking(ThirdChildNode(B))
                                     /* the third node is blinking */

Endrule

Startrule "First and second vacant components ((A, 1) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 3
  Then
    ThirdChildNode(B) <- FirstChildNode(A)
    Blinking (SecondChildNode(B))
                                     /* the second node is blinking */
    Blinking(FirstChildNode(B))
                                     /* the third node is blinking */

Endrule

```

For instance, given a selected expression like  $\sqrt{y^2 + x}$ , and the target-pattern an expression affected by the sub and superscript components, the obtained result is:

$$(y^2 + x)_{\langle vacantelement \rangle}$$

### 4.3.2 Twofold Component Pattern

The selected pattern is composed of two components, and the target-pattern can have two, three, or one component.

1. Transformation ((A,2)(B,2)), (A=B)

When the number of components of both patterns are the same (Number\_parameter(A)=Number\_parameter(B)), and both patterns are the same (RootNode(A) = RootNode(B)). The only possible transformation is to exchange the contents.

```
Startrule "Exchanging contents ((A, 2) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 2
    Number_parameter(B) = 2
    RootNode(A) = RootNode(B)
                                     /* Both patterns are the same */
  Then
    FirstChildNode(B) <- SecondChildNode(A)
                                     /* exchange contents */
    SecondChildNode(B) <- FirstChildNode(A)

Endrule
```

For instance, given the selected expression:  $\frac{x^2}{y^2}$ , and the target-pattern: the fraction, the obtained result is:

$$\frac{y^2}{x^2}$$

2. Transformation ((A,2)(B,2)), (A<>B)

When the number of parameters are equal (Number\_parameter(A) = Number\_parameter(B)), and the patterns are different (RootNode(A) <> RootNode(B)), The possible solution is to assign all contents of the selected pattern to the target one. The following rule fulfills the nodes in the same order (e.g. FirstChildNode(A) is assigned to the FirstChildNode(B)).

```
Startrule "Exchanging operators ((A, 2) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
```

```

    Number_parameter(A) = 2
    Number_parameter(B) = 2
    RootNode(A) <> RootNode(B)
Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- SecondChildNode(A)

Endrule

```

For instance, given the selected expression:  $\frac{x^2}{y^2}$ , and the target-pattern the "under root", the obtained transformation is:  $x^2\sqrt{y^2}$

Once the previous rule has been applied, and if the user now wants to exchange the contents, e.g.  $y^2\sqrt{x^2}$ , the user must select the resulted expression, and he/she can ask for a new transformation. The required rule to be applied is the previous "Exchanging contents (A,2) (B,2)" rule, where the patterns are equal (A=B) and the resulting of the transformation is a permutation of the A pattern's contents.

### 3. Transformation ((A,2)(B,3))

When the number of components of the targeted B pattern is one greater than the number of the A pattern (Number\_parameter(A) = 2 and Number\_parameter(B) = 3), there will be one vacant element (e.g. ThirdChildNode(B)), as expressed in the next rule:

```

Startrule "From two to three components ((A, 2) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 2
    Number_parameter(B) = 3
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- SecondChildNode(A)
    /* the first and second node are completed */
    Blinking(ThirdChildNode(B))
    /* the third node is vacant */

Endrule

```

For instance, given the superscript expression:  $y^2$ , and the target-pattern the "sub-superscript", the obtained result is:  $y^2_{<vacantelement>}$

As we see, the vacant component can be one of the three components of the targeted pattern. There are six possible combinations that can be expressed by rules, for instance:

```

Startrule "From two to three components and second vacant ((A, 2) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 2
    Number_parameter(B) = 3
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    ThirdChildNode(B) <- SecondChildNode(A)
    Blinking(SecondChildNode(B))
                                     /* the second node is vacant */

Endrule

```

```

Startrule "From two to three components and first vacant ((A, 2) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = ''transform_element''
    Number_parameter(A) = 2
    Number_parameter(B) = 3
  Then
    SecondChildNode(B) <- FirstChildNode(A)
                                     /* the first node is copied to the first */
    ThirdChildNode(B) <- SecondChildNode(A)
    Blinking(FirstChildNode(B))
                                     /* the first node is vacant */

Endrule

```

The Knowledge Base can either include here the six possible permutations, or include only the "From two to three components ((A,2) (B,3))" rule, and to process the remaining permutations of the pattern with three components later by other rules, as we see in next sub-section 4.3.3.

It means, once the result is obtained, and if the user now wants to exchange the contents, e.g.  $y_2^{<vacantelement>}$ , the user must select the resulted expression, and he/she can ask for a new transformation. The required rule to be applied is the "Exchanging contents (A,3) (B,3)" rule (see sub-section 4.3.3), where the patterns are equal ( $A=B$ ) and the resulting of the transformation is a permutation of the A pattern's contents.

#### 4. Transformation ((A,2)(B,1))

When the number of the components of the A pattern ( $\text{Number\_parameter}(A)$ ) is one greater than the number of the targeted pattern ( $\text{Number\_parameter}(B)$ ), the user decides which component will be conserved.

```

Startrule "Reducing components ((A, 2) (B, 1))"

```



```

If nature(fragment_1) = "formula"
  action(x) = "transform_element"
  Number_parameter(A) = 2
  Number_parameter(B) = 1
Then
  FirstChildNode(B) <- FirstChildNode(A)

Endrule


Startrule "Reducing components ((A, 2) (B, 1))"
If nature(fragment_1) = "formula"
  action(x) = "transform_element"
  Number_parameter(A) = 2
  Number_parameter(B) = 1
Then
  FirstChildNode(B) <- SecondChildNode(A)

Endrule

```

For instance, given the expression:  $\frac{x^2+1}{y}$ , and being the target-pattern the "square\_root", the obtained results are:  $\sqrt{x^2 + 1}$  and  $\sqrt{y}$

### 4.3.3 Threefold Component Pattern

An example of this kind of pattern is the sub-superscript pattern .

Examples of expressions that can be written using this pattern are: the sub-superscript, made up the base, the subscript and the superscript. For instance:

$$\int_0^1 x dx$$

$$\int_a^b x dx$$

$$\int_{x=1}^{x=100} x dx$$

The target-pattern can have three, two or one component.

1. Transformation ((A,3) (B,3)), (A = B)

When the patterns (RootNode(A) = RootNode(B)) are the same, the transformation can take place by making the permutation of the component positions. The following rules only describe three possible cases. The user will choose one of these possibilities.

```

Startrule "First exchanging contents ((A, 3) (B, 3), (A=B))"
  If nature(fragment_1) = "formula"
    action(x) = ''transform_element''
    Number_parameter(A) = 3
    Number_parameter(B) = 3
    RootNode(A) = RootNode(B)
  Then
    FirstChildNode(B) <- SecondChildNode(A)
    SecondChildNode(B) <- FirstChildNode(A)
    ThirdChildNode(B) <- ThirdChildNode(A)
                                     /* the first combination */

```

```

Endrule

```

```

Startrule "Second exchanging contents ((A, 3) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 3
    RootNode(A) = RootNode(B)
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- ThirdChildNode(A)
    ThirdChildNode(B) <- SecondChildNode(A)
                                     /* the second combination */

```

```

Endrule

```

```

Startrule "Third exchanging contents ((A, 3) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 3
    RootNode(A) = RootNode(B)
  Then
    FirstChildNode(B) <- ThirdChildNode(A)
    SecondChildNode(B) <- FirstChildNode(A)
    ThirdChildNode(B) <- SecondChildNode(A)
                                     /* the third combination */

```

```

Endrule

```

Example, given the "subsuperscript" expression:  $x_y^2$ , some possible transformations are:

$y_x^2$ ,  $x_2^y$ ,  $y_2^x$ , and so on.

2. Transformation  $((A,3) (B,3)), (A \leftrightarrow B)$ 

When the number of components are the same ( $\text{Number\_parameter}(A) = \text{Number\_parameter}(B)$ ) and the patterns are different ( $A \leftrightarrow B$ ), the components are assigned in the same order to the targeted pattern.

```
Startrule "Rename pattern ((A, 3) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 3
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- SecondChildNode(A)
    ThirdChildNode(B) <- ThirdChildNode(A)
    /* Each component is copied in the same order */
Endrule
```

In the case that the component positions are not suited to the user, the above "First exchanging contents  $((A,3) (B,3)), (A=B)$ " rule can be used.

3. Transformation  $((A,3) (B,2))$ 

When the number of parameters of the pattern A is greater than the number of parameters of the targeted pattern B, six possibilities arise. Next, we give the rules corresponding to three of these possibilities.

```
Startrule "First reduction from 3 to 2 components ((A, 3) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 2
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- SecondChildNode(A)
    /* First possibility */
Endrule
```

```
Startrule "Second reduction from 3 to 2 components ((A, 3) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 2
  Then
```

```

FirstChildNode(B) <- SecondChildNode(A)
SecondChildNode(B) <- ThirdChildNode(A)
                                /* Second possibility */

Endrule

Startrule "Third reduction from 3 to 2 components ((A, 3) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 2
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- ThirdChildNode(A)
                                /* Third possibility */

Endrule

```

The other three remaining rules can be similarly obtained.

Example, given the selected expression  $(x + y)_n^3$ , and the target-pattern the "super-script", the obtained result is:  $(x + y)^3$

#### 4. Transformation ((A,3) (B,1))

When the targeted pattern has only one parameter, one of the components of the pattern A is selected. The user can choice one among the three possible cases.

```

Startrule "First reduction from 3 to 1 component ((A, 3) (B, 1))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 1
  Then
    FirstChildNode(B) <- FirstChildNode(A)
                                /* The first possibility */

Endrule

Startrule "Second reduction from 3 to 1 component ((A, 3) (B, 1))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 1
  Then
    FirstChildNode(B) <- SecondChildNode(A)

```

```

/* The second possibility */

Endrule

Startrule "Third reduction from 3 to 1 component ((A, 3) (B, 1))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 1
  Then
    FirstChildNode(B) <- ThirdChildNode(A)
    /* The third possibility */

Endrule

```

Example, given the selected expression:  $(x + y)_n^3$ , and the target-pattern:  $\sqrt{\quad}$ , the obtained transformation can be:  $\sqrt{(x + y)}$ ,  $\sqrt{n}$ ,  $\sqrt{3}$

Table 4.1: Examples of formula transformations

Selected expression	Pattern A	n	C <sub>1</sub>	Expression obtained	Target pattern B	m	C <sub>2</sub>
$\frac{2*x*(x+y)}{5*y}$		2	9.14	$\frac{5*y}{2*x*(x+y)}$		2	9.14
$\frac{2*x*(x+y)}{5*y}$		2	9.14	$5*y\sqrt{2*x*(x+y)}$		2	16.02
$\sqrt{x^4 - x^2 + 1}$		1	12.00	$\int_0^1 x^4 - x^2 + 1 dx$		3	16.00
$\sum_{i=1}^n \frac{n*(n+1)}{2}$		3	30.70	$\left(\frac{n*(n+1)}{2}\right)$		1	5.80

In Table 4.1, we present some examples of transformations, including the complexity of each one. When components of a pattern are only exchanged, the complexity remains the same. The procedure to calculate the complexity of a MathML formula is explained in Chapter 5.

## 4.4 Prefix and Postfix Notation

The Reverse Polish Notation (RPN) was introduced in [42], later it was translated and published in [74]. This notation is widely used, for instance, for rewriting mathematical expressions. This kind of representation is provided by our editor application, under user's request, but displayed in another window.

Table 4.2: Symbols for MathML patterns

MathML pattern	Representative characters
fraction	/
square root	<SQRT>
under root	<RT>
superscript	<SUP>
subscript	<SUB>
sub-superscript	<SUBSUP>
underscript	<UND>
overscript	<OVR>
under-overscript	<UNDOVR>

The contents of the MathML formula are captured into a string, and we change patterns by simple symbols or sequence of characters, as we see in Table 4.2. This string is converted to prefix and postfix forms using the Reverse Polish Notation (RPN) [42][74].

Some examples of rewriting expression in the prefix and postfix form are:

Given the expression  $\frac{a+b}{c}$

Its prefix form is:  $/ + abc$ . Its postfix form is:  $abc + /$

Given the expression:  $\sqrt{\frac{b^2-3*a*b}{9*a^2}}$

Its prefix form is:

$<SQRT>/-<SUP>b2**3ab*9<SUP>a2$

Its postfix form is:

$b2<SUP>3ab**-9a2<SUP>*/<SQRT>$

## 4.5 The Scalable Vector Graphics, SVG Schema

The GAIE inference engine notifies the existent tools or keyboard shortcuts to handle different kind of objects (table, MathML, and SVG), when no information explicitly exists in the main menu or sub-menu.

For instance, figures/images can be moved or resized by holding pressed the *ctrl* key and left mouse button, as shown in Figure 4.7 . No menu option from the tool bar is

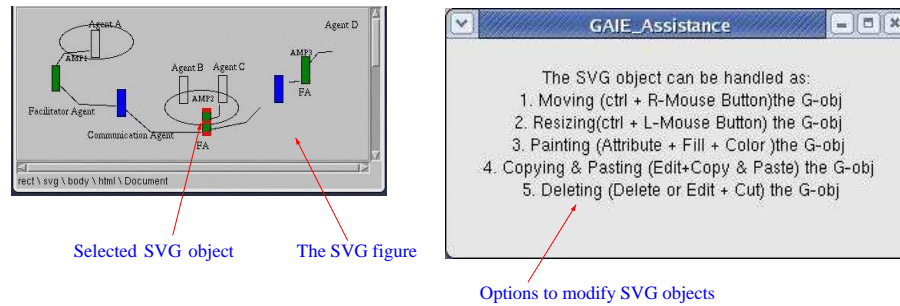


Figure 4.7: Notification of available tools

directly available to produce the same effect. In this case, the "SVG object handling" rule is applied.

```

Startrule "SVG object handling"
  If author(fragment_1) = x
    nature(fragment_1) = "image"
    action(x) = "select_element"
  Then
    announce(x) <- list_tools(SVG_handling)
                                     /* SVG handling commands */

```

Endrule

Similarly, when the components of a specific object, like a table, are modified, (adding, deleting or splitting a row/column) no specific entry on the tool bar is available, thus the GAIE inference engine notifies how to handle this object (see Figure 4.8).

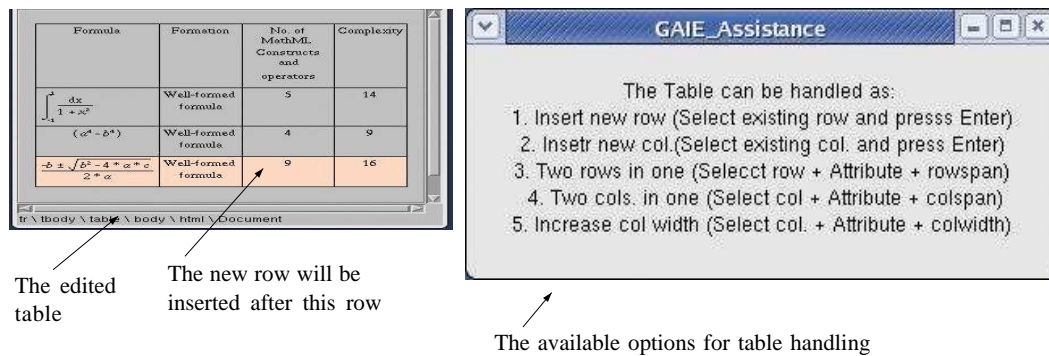


Figure 4.8: The GAIE inference engine informs coauthors about existing tools for tables

The key idea is to notify the possible handling commands that are related to all composed objects, when no type menu is included on the tool bar. The "Table handling" rule is applied. In this way, the production time spent by users is optimized.

```

Startrule "Table handling"
  If author(fragment_1) = x
    nature(fragment_1) = "table"
    action(x) = "select_element"
  Then
    announce(x) <- list_tools(table_handling)
                                /* table handling commands */

Endrule

```

The notifications of keyboard sequences (keyboard shortcuts) allow users to more independently produce the complex objects without disturbing their colleagues.

## 4.6 Conclusion

The MathML language is widely used, for instance, Amaya [24], Mathematica [41], ActiveMath [50], Internet Explorer<sup>1</sup>, but none of these applications validate if the produced expression is well-formed.

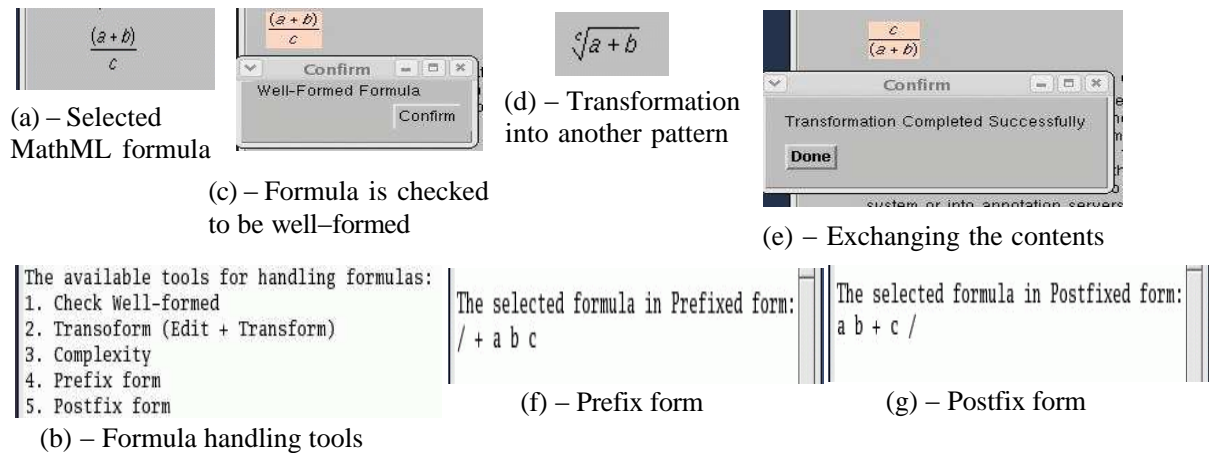


Figure 4.9: Handling mathematical expressions

We integrate into our authoring application, an assistant to help users during the production of a formula (see Figure 4.9).

- When a formula is finished or selected the system notifies if the formula is well-formed (a)(c).
- Reusable\_pattern is also possible by transforming a selected pattern expression into another targeted pattern (d)(e).

<sup>1</sup><http://www.microsoft.com/windows/ie/>



- A MathML formula can be rewritten into prefixed or postfix forms, and displayed on an external window (f)(g).

Once a formula is updated, the system notifies if it is well-formed and displays other functions that can be applied to it (Figure 4.9(b)). The complexity of a formula is automatically computed, and it is used to evaluate the cooperative production (Chapter 5).

The MathML transformation tool is the starting point of the construction of a catalog of the reusable resources (formulas, SVG graphics, tables, etc.) for the cooperative group.



# Chapter 5

## Evaluation of Cooperative Production

The Group Awareness Inference Engine (GAIE) was designed and implemented to enhance awareness between experts and beginners and improve the coordination. In this chapter, we report the way in which our group awareness system evaluates the cooperative production and provides assistance in case of production trouble. The related works are also presented.

### 5.1 Introduction

Due to the different structure handled by our authoring application, it is possible to distinguish the nature of the production (annotation, text, headings, paragraphs, lists, hypertext links [35], MathML [8], and SVG schemes [54]) and the writing actions executed on it (undo/redo, copy/cut/paste, etc.)

In order to distinguish the executed writing actions, we develop a Distributed Event Management Service (DEMS) [12]. The DEMS filters the information needed by the GAIE inference engine to deduce important facts.

For instance, a user wants to produce an elaborated formula, but he/she spends a lot of time to complete it. The inference engine deduces that he/she has some production troubles, then it suggests him/her some suited functions (Chapter 4) included in the authoring application to improve his/her production, and the inference engine provides him/her information to contact an expert (section 5.3.1).

Due to the different roles played by users within the cooperative environment, the evaluation of the collaborative production is necessary. Taking into account the evaluation of the user's productiveness the GAIE inference engine can qualify a user as an expert during a session, so it is a dynamic evaluation.

## 5.2 Evaluation of MathML Expressions

The evaluation refers to the analysis concerning: a) the kind of elements included in the shared document: text, tables, links, MathML, SVG schemas. For instance, a figure is defined by the vertical and horizontal alignment, its appearance, etc. b) the number and the complexity of these elements, and c) the elapsed time to produce them.

In the case of the MathML production, the evaluation is based on: a) the number of *well-formed* formulas produced and their complexity, b) the elapsed time to produce them, and c) the performed actions during their production (undo/redo, copy/cut/paste, etc.).

As we see in Chapter 4, a formula is *well-formed*, if all elements composing a MathML pattern are completed and each element, in turn is a well-formed MathML expression too.

Once a formula is concluded, the GAIE inference engine verifies if it is well-formed. Afterwards, its complexity is calculated.

### 5.2.1 The Complexity of a Formula

The complexity of a formula depends on the way by which a formula is created ("Type Menu", or "MathML icon") (Chapter 4), the steps taken, and the elapsed time to complete it: number of times the "type menus" are selected, as well as the math dialog box, and the math palette are used.

**a) "Type Menu".** A formula can be created by selecting in the "Types Menu", the "Math" option, followed by the "New formula" option.

Basic expressions are completed from the standard input (keyboard), producing a sequence of symbols: identifier/constant, unary/binary operator, and delimiters.

The assigned values to this kind of expressions are shown in Table 5.1, from rules #1 to #6. Thus,  $(x + y - z)$  has a complexity of 4.50.

**b) "MathML icon".** A formula can be created by clicking the "MathML icon" on the tool bar, followed by the selection of the first "Math" button in the "MathML box" (see Figure 5.1).

$$\sqrt{x^2 - 2 * x * y + y^2} \quad (5.1)$$

A MathML pattern can be characterized by the number of components to be fulfilled, like those shown in Table 5.1, rules from #7 to #14:

1. Single Component (square root). The complexity is equal to the complexity of the single component base expression, multiplied by 1.5.

For instance, giving the expression 5.1, and applying the rules #1, #2, #3, #4, #5, #9 and #10 from Table 5.1, the resultant complexity is 17.40.

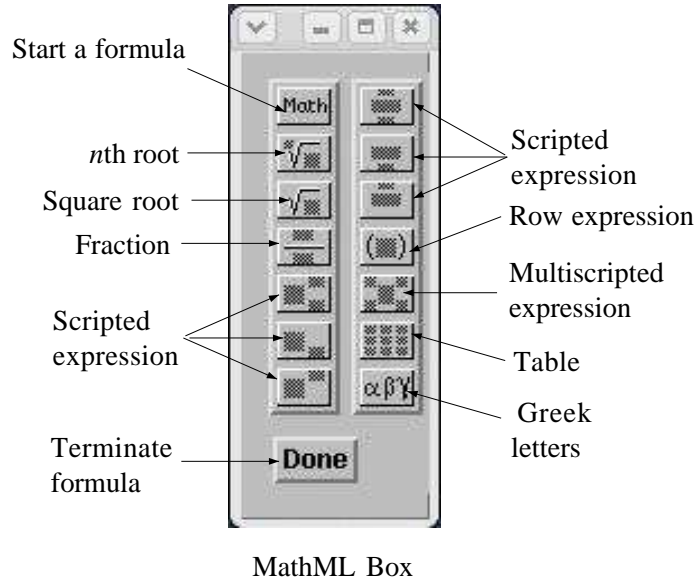


Figure 5.1: MathML patterns

2. Twofold Component. Examples of this kind of pattern are: under root, superscript, subscript, and fraction. The complexity depends on the basic-expression associated to the pattern as we see in Table 5.1, rules #7, #8, #9, and #11.

For example, the same number of steps are involved in producing a fraction:  $\frac{10}{5}$  (complexity = 2.5), or an exponent:  $10^5$  (complexity = 3.0), although, we think that the latter expression is more complex than the first one (Table 5.1, rules #7 and #9).

The complexity of the following expression is 23.58.

$$\sqrt[x]{x^2 - 2 * x * y + y^2}$$

3. Threefold Component (Sub-Superscripts, and Under-Overscripts). The complexity of these patterns is related/associated to the complexity of the "z" base component of the patterns  $\int, \sum z, \prod z$ , (rules #12, #13, and #14 from Table 5.1 and the complexity of the superscript or the overscript expression.

The computed complexity of expression 5.2 is 10.00 as a result of applying the rules: #1, #2, #5, #6, #9 and #12 from Table 5.1.

$$\int_1^4 3 * x^2 dx \tag{5.2}$$

The formula 5.3 has been also produced using sub-superscript pattern, whose base component includes a fraction and the denominator contains a superscript expression.

Table 5.1: Complexity of mathematical expressions

MathML pattern	Rule number	Complexity
constant	1	1.0
variable	2	1.5
$x + y$	3	complex (x) + complex (y)
$x - y$	4	complex (x) + complex (y)
$x * y$	5	complex (x) + complex (y) + 0.3
$x = y$	6	complex (x) + complex (y)
$\frac{x}{y}$	7	complex (x) + complex (y) + 0.5
$x_y$	8	complex (x) + complex (y) + 0.8
$x^y$	9	complex (x) + complex (y) + 1.0
$\sqrt{x}$	10	complex (x) * 1.5
$x\sqrt{y}$	11	complex (x) + complex (y) * 1.8
$\int_x^y z dz$	12	(complex (x) + complex (y)) * complex (z)
$\sum_x^y z$	13	((complex (x) + complex (y)) * complex (z)) + 1.5
$\prod_x^y z$	14	complex (x) * complex (y) * complex (z)

This superscript expression carries two more superscript expressions separated by a minus. The complexity of formula 5.3 is 20.00 as a result of applying rules: #1, #2, #4, #7, #9 and #12 from Table 5.1.

$$\int_0^1 \frac{x dx}{(a^2 - x^2)^2} \quad (5.3)$$

Using the under-overscript pattern, the base can be completed by functions like: "Σ" or "Π". Thus, we think that the complexity for each one can be different. The expression 5.4 has a complexity of 30.70, by applying the rules: #1, #2, #3, #5, #6, #7, and #13 from Table 5.1.

$$\sum_{i=1}^n i = \frac{n * (n + 1)}{2} \quad (5.4)$$

The complexity of a formula is calculated by the following rule.

```
Startrule "Computing complexity"
  If author(fragment_1) = x
    role(x) = "Writer"
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    evaluate_wff(fragment_1) = "true" /* formula is wff */
  Then
```

```

announce(x) <- computed_complexity(fragment_1)
/* The complexity of the formula is calculated */
Endrule

```

A user can ask the GAIE inference engine to calculate the complexity of a produced MathML formula. In this case, the following rule is applied:

```

Startrule "Computing complexity on request"
If action(x) = "select_formula"
  nature(fragment_1) = "formula"
Then
  announce(x) <- computed_complexity(fragment_1)
  /* The complexity of the formula is calculated */
Endrule

```

Table 5.1 establishes the complexity of basic expressions, and these patterns are given as input data to the system, thus, it is possible to change these values without modifying the code of the GAIE inference engine.

In addition to determine a well-formed formula and calculate its complexity, other factors are also considered to evaluate the user's production: the undo/reverted actions, the elapsed time and the change in size of the produced document, as we see in the next subsection.

### 5.2.2 Undo/redo Action

We performed an experiment with 10 users, 5 of them had known how to write elaborated large documents (expert). We asked them to reproduce some formulas, like binomials, integrals, and quadratic ones.

For instance, the average number of performed actions of the experimented user, to produce the expression 5.5, were 20, and the reverted actions were 2. Whereas, beginners reproduced the same expression, having an average number of 38, while the reverted actions were 18.

The counted actions are: starting of the formula (selection of "maths" button from MathML box), insertion of each pattern (square root, superscript), positioning the cursor in the pattern, producing the greek letters from math palette (see Figure 5.2), and insertion of each number, identifier, and operator from the standard input (keyboard).

$$\frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 * a} \quad (5.5)$$

Normally, experimented users revert minimum actions to perform their task.

In the case of the formula 5.6, beginners executed an average of 9 reverted actions, out of 14 actions.

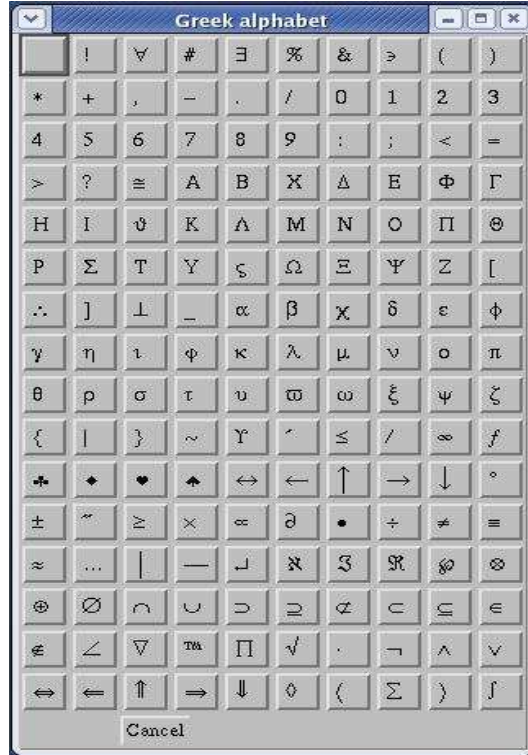


Figure 5.2: Greek letters

$$a^2 - b^2 \tag{5.6}$$

The following rule is applied to register the undo/redo actions, when a user produces a MathML formula:

```

Startrule "Actions-time-size"
  If action(x) = "start_formula"
    nature(fragment_1) = "formula"
  Then
    total_actions(x) <- start_count_total_actions(x)
                        /* actions are counted */
    undo_actions(x) <- start_count_undo_actions(x)
                      /* reverted actions are counted */
    initial_time(fragment_1) <- register_current_time(fragment_1)
                              /* the time is recorded */
    initial_size(document) <- register_initial_size(document)
                              /* the size is recorded */
Endrule

```

When a user starts a MathML formula, the number of reverted actions, the current system time, as well as the size of the document are registered.



When a user reverts more than half of his/her actions, he/she is notified of presence of experts with whom the former can communicate with. The following rule is applied:

```

Startrule "Reverted actions"
  If author(fragment_1) = x
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    undo_actions(x) > total_actions / 2
      /* author reverts more than half of performed actions */
    status_open(authors_definition_data_base) = "true"
  Then
    announce(x) <- "MathML_expert_exists"

Endrule

```

When the users produce a formula, their performed actions are observed, if they are in trouble, the system proposes them to contact experts using synchronous communication.

The *well-formed* formulas produced within a shared document during a session are counted by the following rule:

```

Startrule "Counting the well-formed formulas"
  If nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    evaluate_wff(fragment_1) = "true" /* formula is wff */
  Then
    summary_wff(x) <- summary_wff(x) + 1
      /* Well-formed formulas are counted */

Endrule

```

At the time of creation of a shared document, the value for the *well-formed* formulas is initialized as zero. When a *well-formed* formula is produced, the value is incremented by 1, as it is done by "*Counting the well-formed formulas*" rule.

Depending on the number of produced well-formed expressions and their complexity, the GAIE inference engine can classify a user as a possible expert on writing expressions.

### 5.2.3 Elapsed\_Time

The time during which a user writes a well-formed mathematical expression in one session, without performing other activities is the "elapsed time". This time is calculated by taking the difference between the registered time when a user starts to produce a formula and when he/she completes it (clicking the "Done" button of MathML box)(see Figure 5.1).

The GAIE inference engine memorizes the time when the formula is concluded. The following rule is applied:

```

Startrule "Final time-Final size"
  If author(fragment_1) = x
    role(x) = "Writer"
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
  Then
    final_time(x) <- register_current_time(x)
                        /* the time is recorded */
    final_size(fragment_1) <- register_final_size(fragment_1)
                        /* the size is recorded */
    elapsed_time(x) <- final_time(x) - initial_time(x)
                        /* the elapsed time is computed */
    calculate_change_size(fragment_1) <-
                        final_size(fragment_1) - initial_size(fragment_1)
                        /* the change in size is computed */

Endrule

```

From the group of 5 experts and 5 beginners, we studied that initially the beginners were spending more time to insert mathematical patterns (fractions, exponents, subscripts, etc.). The duration was decreased when the expertise was grown to insert data and produce elaborated MathML expressions. The observed average spent time was:

```

Production time (in seconds)=
  a*(number of MathML patterns)+
  b*(number of greek letters)+
  c*(characters from standard input)+
  (elapsed time for the selection of
   menus, MathML box, math palette, and
   position to insert characters)
where a=b=5, c=2.

```

These values were obtained according to the experimentation carried out.

We observed that experts take less than 5 minutes (average) to produce a formula containing mathematical patterns and greek letters (e.g. quadratic formula). Therefore, when a user spends more than 10 minutes to produce a formula, whose complexity is less than 16, he/she is informed about the existence of an expert. The following rule applies:

```

Startrule "Large elapsed time"
  If author(fragment_1) = x
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    elapsed_time(x) > 10      /* time in minutes */
    computed_complexity(fragment_1) < 16
    status_open(authors_definition_data_base) = "true"

```

Table 5.2: Elapsed Time to produce MathML expressions (in seconds)

MathML Expressions	Comp.	Time spent by experts in seconds					Ave.	Time spent by beginners in seconds					Ave.
		E1	E2	E3	E4	E5		-	N1	N2	N3	N4	
-	-	E1	E2	E3	E4	E5	-	N1	N2	N3	N4	N5	-
$a^2 - b^2$	8.0	14	15	18	14	15	15	124	90	80	65	150	102
$(x - y)^4 = x^4 + 4 * x^3 * y + 6 * x^2 * y^2 - 4 * x * y^3 + y^4$	32.8	167	158	140	155	149	154	580	719	246	363	501	482
$\frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 * a}$	16.6	177	40	96	83	91	97	734	617	540	616	617	625
$\sum_{n=0}^{\infty} a_n * t^n$	16.2	118	70	80	91	86	89	526	430	152	420	223	350
$\pi \left( 2 * a_0^2 + \sum_{n=1}^{n=N} (a_n^2 + a_n^2) \right) *$	56.1	141	100	120	98	107	117	619	465	326	254	297	372
$\int_{-\pi}^{\pi} f_N^2(x) dx$	26.0	135	70	104	109	99	103	425	356	183	265	225	290
$n! \sim \sqrt{2 * \pi} * n^{(n+\frac{1}{2})} e^{-n}$	21.7	93	65	73	79	89	80	360	282	125	124	221	222
$e^{-C*x} * \frac{1}{x} * \prod_{n=1}^{+\infty} \frac{e^{\frac{x}{n}}}{1 + \frac{x}{n}}$	50.6	100	102	121	113	111	109	585	425	747	494	280	506
$\int_a^b (1 - t)^{x+y-1} \left( \frac{t}{1-t} \right)^x dt$	45.0	122	85	104	117	113	106	495	430	670	398	722	543

Then

```
announce(x) <- MathML_expert_exists"
```

Endrule

The results obtained in the experimentation made for producing MathML expressions is shown in Table 5.2. The produced formulas, their complexities, and the elapsed time by each expert and beginner are presented. The average spent time of experts and beginners is also computed. For instance, to produce a quadratic formula (shown in boldface), experts spent 97 seconds (1.62 minutes) on average while beginners took 625 seconds (10.42 minutes).

## 5.2.4 Evolution of the Production

The size of the shared document is registered when: 1) a user starts to produce a formula, and 2) he/she completes a well-formed version. The difference between registered sizes gives the variation in the size of the production in bytes as we see in the "Final time-Final size" rule in section 5.2.3.

Figure 5.3 shows the steps taken by a user to write a quadratic formula. The increase in size after the insertion of each component is recorded. The fraction pattern causes an increase of 45 bytes, whereas the root, and the superscript cause an increase of 45, and 67 bytes respectively. Each time a new pattern is included, we observe (see Figure 5.4, a, b,

Element	Formula	Increase in size	Difference
paragraph		9	9
math		66	57
fraction	$\frac{1}{1}$	111	45
operator	$-$	114	3
identifier	$-b$	156	42
operator	$-b \pm$	183	27
root	$\frac{-b \pm \sqrt{1}}{1}$	228	45
superscript	$\frac{-b \pm \sqrt{1^1}}{1}$	295	67
identifier	$\frac{-b \pm \sqrt{b^1}}{1}$	282	-13
number	$\frac{-b \pm \sqrt{b^2}}{1}$	285	3
operator	$\frac{-b \pm \sqrt{b^2 -}}{1}$	304	19
number	$\frac{-b \pm \sqrt{b^2 - 4}}{1}$	323	19
operator	$\frac{-b \pm \sqrt{b^2 - 4 *}}{1}$	342	19
identifier	$\frac{-b \pm \sqrt{b^2 - 4 * a}}{1}$	361	19
operator	$\frac{-b \pm \sqrt{b^2 - 4 * a *}}{1}$	380	19
identifier	$\frac{-b \pm \sqrt{b^2 - 4 * a * c}}{1}$	399	19
number	$\frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2}$	402	3
operator	$\frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 *}$	444	42
identifier	$\frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 * a}$	461	17

Figure 5.3: Observations of the formula size increase

and c) an increase of the size. During the time, the components of the pattern are empty, the size of the document remains unchanged.

When an expert develops a MathML formula, the document's size gradually increases (see Figure 5.4), whereas, the change in size was very slow when a beginner produced it. The time is taken on X-axis with an interval of 10 seconds, and the increase in size is taken on Y-axis with an interval of 50 bytes.

In our study, the average increased size is approximately 50 bytes per minute in case of expert's production whereas, a beginner produces less than 15 bytes per minute.

Thus, when the difference in size of the production is less than 10 bytes per minute, the user is asked if he/she is in trouble. We exempt the possibility of inactive or idle sessions. This is performed by the rule:

```
Startrule "Change in size"
```

```

If author(fragment_1) = x
  nature(fragment_1) = "formula"
  update(fragment_1) = "true"
  caculate_change_size(fragment_1) < 10 /* size in bytes */
  elapsed_time(x) > 1 /* time in minutes */
  status_open(authors_definition_data_base) = "true"
Then
  announce(x) <- "MathML_expert_exists"

```

Endrule

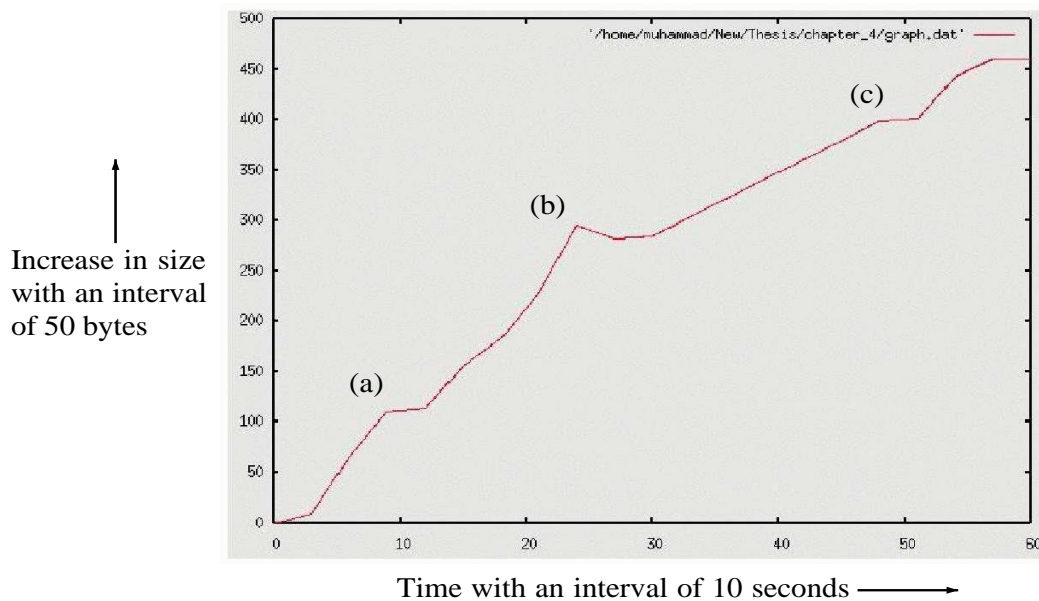


Figure 5.4: Document size increases with time

We observed the following behavior of the beginners when they were asked to produce a quadratic expression 5.5 in our authoring application:

- They wrote formulas without changing the schema XHTML or SVG to MathML,
- They spent reasonable time (3 to 5 minutes) to find the MathML box or the menus to produce the formula. They preferred the use of MathML box to "Type menu" to produce different patterns,
- They spent reasonable time (3 to 5 minutes) to insert greek letters,
- They spent 10 to 12 minutes to produce the formula,
- They preferred the use of keyboard to the MathML palette to produce operators (e.g., +, -, \*, !, ~, <, >),

- They did not care about the number of components required by the MathML patterns.

### 5.3 Presence Awareness and Assistance

"The information about the presence of collaborators, their attitudes during a communication session, when they are busy and how frequently they produce objects of a particular nature" [9] is defined as **presence/social awareness**.

The classical queries about social awareness are: What is a user doing? Is he/she busy talking with someone? Can he/she be disturbed? Does he/she want to communicate with someone? Can he/she assist his/her colleagues? .... The improvement of social awareness entails better coordination of the group activities [43].

Let us consider a scenario: "Stephan normally produces texts. Due to the document requirements, he has to write a quadratic formula and he has to submit this document within a short period of time. Due to lack of information, he does not know how he can start an expression and produce it. After many attempts he could not succeed to complete his task". At this stage, it is nice to inform "Stephan" the availability of expert partners in this respect.

The coauthor in trouble is notified about the presence of experts as shown in Figure 5.5. The notification box offers three buttons: - "Information" to get data about expert like: login name, working site, and availability; - "Contact" to synchronously communicate with them; and - "Cancel" to close the notification box.



Figure 5.5: Notification of the presence of experts and their availability

Suppose, Stephan is interested to communicate with Paul, who commonly produces mathematical expressions. Hence, Stephan may start a communication session with him, and ask how to produce a quadratic expression, as we see in Figure 5.6. By means of the synchronous communication, offered by our awareness system, Paul explains Stephan step by step how to do it.

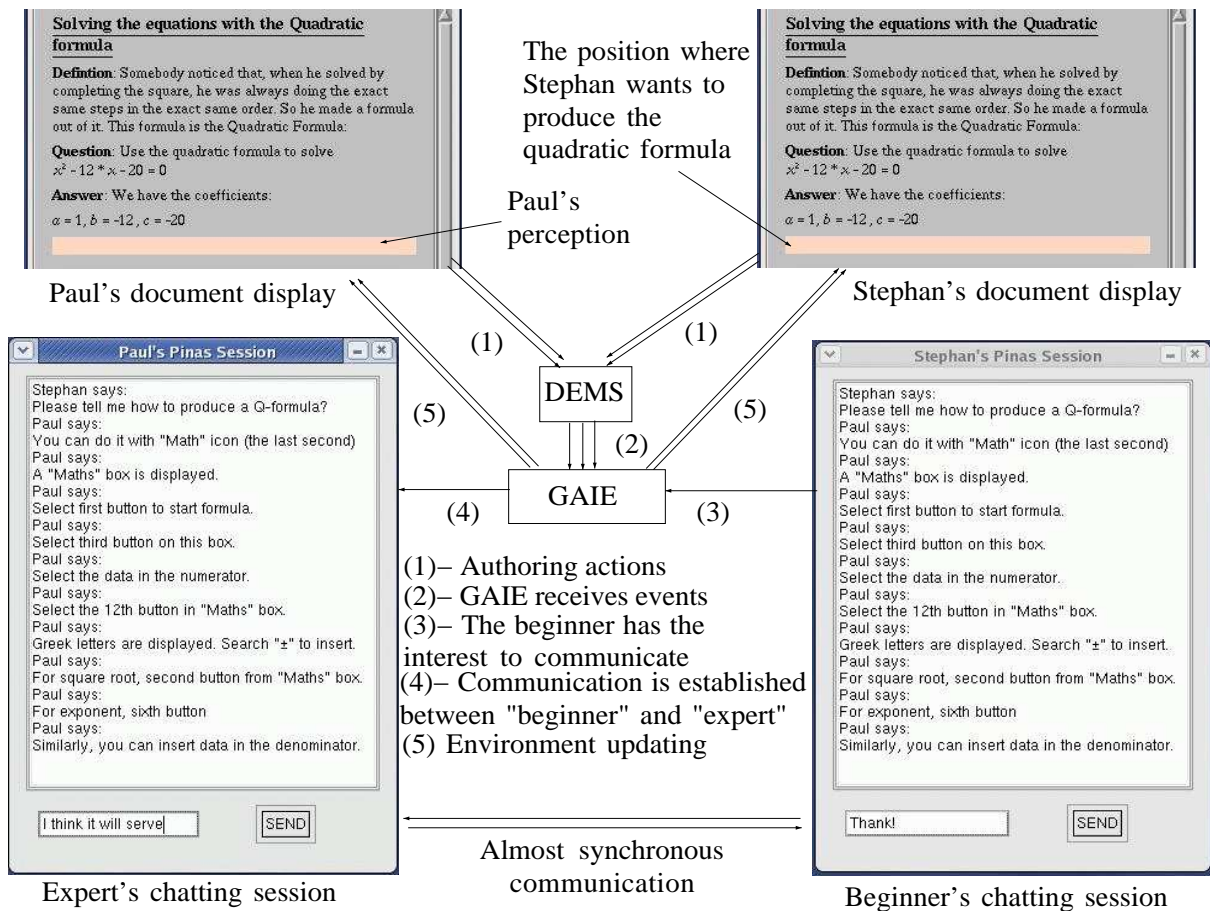


Figure 5.6: The discussion between an expert and a beginner

Once Stephan finishes the formula, he can send its position within the shared document, by selecting the formula. It means, a) the DEMS service captures the selection event and recovers the unique identifier(s) of the selected section(s): "focus of discussion"; b) using this unique identifier, the GAIE inference engine displays the "focus of discussion" into all concerned coauthor's environments.

When the nature of the fragment that a user produces, does not tailor with his/her common production, he/she is informed about the existence of experts. The "*Presence of MathML expert*" rule carries out this task.

```
Startrule "Presence of MathML expert"
```

```
  If author(fragment_1) = x
    role(x) = "Writer"
    status_open(authors_definition_data_base) = "true"
      /* experts in producing MathML elements exist in the database */
    expertise(x) <> "MathML"
      /* 'x' has not expertise to produce MathML components */
    action(x) = ''start_formula''
```

```

Then
  announce(x) <- "MathML_expert_exists"

```

```

Endrule

```

The notification about the immediate user's proximity improves the presence awareness, avoiding individual attitude.

In the following, we explain how to define experts.

### 5.3.1 Deduction of Experts

A user who produces elaborated figures, well-formed formulas, and continuously adds different kind of elements, can be considered as an expert. He/she avoids lost of recent changes by saving the document after a gradual period of time, or certain number of characters, or after addition of a figure, or concluding a formula, or a section.

By contrast, beginners take a lot of time to perform a simple task doing irrelevant and unnecessary actions because they do not know how to proceed.

When a coauthor produces five well-formed formulas in the same session (dynamic evaluation) and the complexity of each one is greater than 16, he/she can be considered a writing mathematical expert. The following rule is applied:

```

Startrule "Defining expertise on writing formulas"
  If author(fragment_1) = x
    role(x) = "Writer"
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    complexity(fragment_1) > 16 /* from Table 4.1 */
    summary_wff(formula) > 5 /* formulas are counted */
    status_open(authors_definition_data_base) = "true"
      /* author's definition file is open */
  Then
    author_definition(x) <== "expert_MathML"
      /*'x' is defined as an expert in writing mathematical formula */
Endrule

```

The expertise of a user is saved in the user definition database, as well as, his/her login, user name, working site, and the availability to communicate with.

Precise information of the ad hoc expert is a suitable information, in case of trouble.

## 5.4 Some Related Work

Analyzing the awareness functionalities provided by some representative Expertise Recommending systems (ERs), we can observe how they provide more or less elaborated functions (see Table 4.3):



- Expertise Recommender (ER) [46][47][48][49] assists the user in trouble who makes a request by giving the area for expertise (e.g. technical support, programming development support), the choice from social network (e.g. students, professors, advisers) and the error code produced by the compiler. The programmers using the ER system, follow the "Line 10 Rule": contacting the member who recently modified the code and asking for help, or the user in trouble performs multiple queries over the support using the indications, customer or program module involved.

The request in order to search experts is made through a dialog box, selecting the "Topic Areas", the "Social Network" and the "Request Code" as shown in Figure 5.7. Users have the possibility to access the database of experts, and know their status. In this way, users can contact these experts, taking into account the topic area and "social network" level.



Figure 5.7: Searching expertise with Expertise Recommender (ER)

- The Expertise Finder (EF) [34] uses the record of the organization and the knowledge of the people to recommend the expertise. It provides a list of the documents and the experts with their contact numbers. The EF system fulfills the request of a user in trouble following two ways: 1) to provide him/her a list of documents which he/she can consult to solve his/her problem, 2) to ask help from experts within the organization.
- Multistrategy Error Detection and Discovery (MEDD) [64] localizes errors in a beginner's program. The main components of the system are: Input (the beginner program), definition of the knowledge level errors and classification of the beginner program with respect to defined errors. Thus, MEDD determines the committed errors in the code and prepares a library of bugs. Using this library, the beginners are informed to rectify errors in the Prolog programs.
- Dynamic Expertise Modeling from Organizational Information Resources (DEMOIR) [72] is a framework which develops and tests expertise modeling algorithms. It recommends the documents and the experts that are locally or remotely available with whom a user in trouble, can contact.

- The developed awareness functionalities in AllianceWeb evaluate the cooperative production on the basis of a dynamic criteria to analyze the mathematical production of a user in order to deduce his/her expertise.

The existence of an expert is notified to beginners when they encounter any problem during the production. The beginners can launch the quasi synchronous communication service to discuss their issues with experts.

Table 5.3: Awareness in Expertise Recommender Systems (ERs)

Expertise recommender systems	Notification of user's expertise	Evaluation criteria for the deduction of expertise	User Assistance
EF	No	Daily activities, human references, problem solving frequency, etc.	Recommendation of some documents or experts
ER	No	Problem solving frequency	Recommendation of experts with a topic area and social network level
MEDD	Not cooperative	Verification of the program code produced by a beginner with some standard algorithms to find bugs in it	Error detection in the code and giving the proper solution
DEMOIR	No	Human references, Information collected from Web resources	Recommendation of experts and frequently consulted documents
GAIE/AllianceWeb	The user in trouble is notified about the presence of experts	Evaluation of the cooperative production on dynamic criteria	Notification of keyboard shortcuts, evaluation of Math expressions to be well-formed, reusing the components of produced formulas

## 5.5 Conclusions

In order to provide a friendly working environment to users, it is necessary to inform them the productiveness of their colleagues.

The productivity of users on writing mathematical formulas can be determined on predefined criteria: Do they produce well-formed formulas?, what is the complexity of each formula? how much does it elapse? how many actions do they perform and revert? how does the size of the production increase? By means of this evaluation, the GAIE inference engine defines a user as a possible expert. The evaluation criteria can be modified at any time in the rule base interpreted by the inference engine.

The inference engine enhances presence/social awareness by notifying the presence of experts to users in trouble. Users can communicate with experts to ask for assistance and hence, they can coordinate their activities.

In future, we intend to establish a criteria to evaluate the graphical production: SVG graphics. For this purpose, the history of the different versions of the shared documents will be necessary, in order to determine the evolution of the graphical objects. In addition, we will take into account several components composing a figure (circle, line, rectangular, etc.), their attributes (color, height, and width), and their relative positions (alignment and inclusion).



# Chapter 6

## Conclusions

The principal contribution of this research is the design and the development of a group awareness system to support cooperative productions/interactions in an efficient way. We validate our approach into a Web-based cooperative authoring application: AllianceWeb.

Using AllianceWeb, a group of collaborators, working at different moments and from different remote sites, are able to produce elaborated and shared Web documents in a consistent way. Taking into account the characteristics of this kind of cooperation, group awareness functionalities constitutes a key issue.

We integrate our awareness functionalities into the AllianceWeb as a seamless system. These functions are developed following Artificial Intelligence principles, particularly those of the Knowledge Based Systems. Thus we designed and implemented a Group Awareness Inference Engine (GAIE).

The functioning principles of the GAIE inference engine are: a) the retrieval of editing events produced during the work session (facts used by rules); b) defining/managing of a Knowledge Base to deduce new interesting knowledge; and c) diffusing and concisely presenting the inferred knowledge to the concerned coauthors. These above principles allow coauthors: 1) to adapt their working environment, and 2) to efficiently produce documents and interact with each other.

All conditions expressed in the rules of the Knowledge Base (KB) are totally independent of our awareness system because a rule is a simple data. The KB can be modified, adapted, and (re)organized, if it is required. Thus, it is not necessary to modify the implemented generic GAIE inference engine, each time the KB has to be adapted to a new situation. Altogether, our approach meet quality advantages of the knowledge based systems.

In addition, a dedicated assistant has been included to provide useful help to coauthors. This assistant allows coauthors to handle mathematical expressions, graphical objects, and tables. For instance, a coauthor can check if the produced formula is well-formed, convert it into prefix and postfix forms, transform it from one presentation to another. In case of graphical objects, the assistant notifies how to resize, and relocate these objects. In the case of a table, instructions on how to insert or split a column/row are notified.

Our contributions are provided in three main lines implemented and validated in the

AllianceWeb cooperative authoring tool: 1) the context-based communication to allow coauthors to precise the "*work focus*" under discussion, 2) the consistency of the cooperative production to allow users to be aware of the interrelated sections of the document ("*work proximity*"), 3) the cooperative production evaluation to deduce/qualify which users are experts, in order to determine coauthors who need assistance and coauthors who may efficiently help others.

**1. Context-Based Communication.** One of the principles, on which our coauthoring editor works, is the document's structure. Based on this structure, a document is partitioned in fragments. Each fragment (e.g. a figure, a table, a paragraph) is identified by a unique document identifier, id. This label is used to precise the "*work focus*": a whole or partial object within the shared document, on which a coauthor wants to center the discussion.

A "*work focus*" is dynamically determined by the GAIE inference engine, and automatically transmitted, distributed and updated in the environment of all concerned coauthors. Having present the "*work focus*" on the coauthor's display, the time to reach a matter of discussion is reduced.

By means of the quasi synchronous communication provided by our awareness system, the collaborators can ask for assistance and discuss about the integrated modifications, production problems, errors, and particular objects within the shared document.

Contrarily, an asynchronous communication requires additional work for sending, receiving messages and attaching large complex information.

Out of any kind of communication (synchronous/asynchronous), it is important to deduce the "*work focus*" under discussion, and to center in it, as quickly as possible.

Thanks to the fact that our cooperative platform (PIÑAS and DEMS) efficiently manages cooperative applications (e.g. AllianceWeb) and non-cooperative ones (e.g. the synchronous communication service), collaborators can simultaneously use the coauthoring editor and communicate with their colleague(s).

The *context-based communication* is similar to a face-to-face interaction, allowing coauthors to coordinate their activities, reach on agreement concerning the cooperative production, and assist each other to solve a problem that appears in such an elaborated cooperative environment.

**2. The Consistency of the Cooperative Production.** The GAIE engine is designed to be able to take into account some relations that may exist between two or several shared objects (e.g. a figure, a table, a formula or a text-part). Thus, if an object (e.g. a figure) is referred by another object (e.g. a text-part), and if each one is currently produced by different coauthors, (e.g. the figure is modified by coauthor "x" and its explanation is produced by a coauthor "y"), these objects must be perceived as inter-related units, and the GAIE inference engine establishes the "*work proximity*" and signals this fact to concerned coauthors.

As coauthors have the possibility to negotiate changes made by others on the related objects, notifications about the *work proximity* allow to ensure the consistency of the

cooperative production, and to come to an agreement, using the offered *context-based communication* service.

**Future work.** We think that it is possible to extend the concept of the "work proximity" to the "cross-reference" among inter-document resources of the shared document base. In this way, it will be extended the consistency of the shared production to the whole document base.

**3. Evaluation of the Cooperative Production.** The evaluation of the cooperative production has been explored. In our studied case, we analyzed the production of mathematical formulas, taking into account the following criteria: "*Is it a well-formed formula? What is its associated complexity? How long is the elapsed time to produce it? How many undo/redo actions are performed during its production? How many well-formed formulas are produced during a work-session?*"

This evaluation is used to determine the productivity skills of a coauthor in order to classify him as a possible expert for the production of MathML formulas. Collaborators who are in trouble in the production of elaborated formulas, often this is the case of beginners, are notified about the coauthors' expertise in this domain.

In order to make coauthors autonomous, an assistant is provided by our awareness system to help users to process specific objects like formulas, tables, and figures. This facility allows authors to continue their production without disturbing their colleague(s).

**Future work.** We will extend our research in group awareness, introducing new criteria to evaluate the production of complex objects like multimedia, figures, and images. For this purpose, the history of the different versions of the shared document has to be supported in order to know in how many sessions the complex objects are completed.

For instance, the evaluation of the graphical production must take into account: a) the great variety of included components: circle, line, ellipse, text fragment, etc.; b) their attributes: color, height, and width; c) their relative positions; d) the elapsed time to produce it.

Since a graphical object is completed in more than one session, currently the evaluation of the graphical production is not possible, due to the unavailable information about the history of the shared document in our authoring application. That is one of our future research goal.





# Appendix A

## Syntax of the GAIE Language

### A) Notation

The symbols `::=`, `[`, `]`, `|` are part of the Backus-Norm Form, as follows:

The symbols `::=`, `{[}`, `]`, `|` are part of the Backus-Norm Form, as follows:

`L ::= R` The syntax of L is defined by R

`[ X ]` An optional item

`X | Y` An item from one of the syntactic categories X or Y

`X*` An item of the syntactic category X is repeated one or more times

- We write the predefined terminal symbols that are part of the GAIE language in boldface e.g. **Startrule**, **If**, **Then**, **Endrule**.
- We write user-defined terminal symbols in italic. They are always atomic symbols
- We write non-terminal symbols in normal type face, e.g. rule
- A comment is enclosed by the symbols: `/* */`, e.g. `/* it is a comment */`
- A constant symbol or a sequence of characters are enclosed by the symbols: `" "`, e.g. `"expert_MathML"`
- A variable symbol is represented by an identifier

**B) Syntax**

```

rule          ::= Startrule id_rule
                If premise_rule*
                Then
                    action_rule*
                Endrule
id_rule       ::= "Sequence of Characters"
premise_rule  ::= condition [/* comments */]
condition     ::= function_id(argument) relational_operator value
argument      ::= identifier
relational_operator ::= = | < | > | <>
value         ::= identifier | constant | expression
expression    ::= expression op_arit expression |
                numeric | Num_function_id (argument)
op_arit       ::= + | - | * | /
action_rule   ::= action_id ( argument ) affectation_op value
affectation_op ::= <- | <==

```

The affectation operator " $<-$ " stands for a function of 1:1

e.g. `count (x) <- count (x) + 1`. It means that the `count(x)` is updated.

The affectation operator " $<==$ " stands for a function of 1:N,

e.g. `author_definition (x) <== "expert_MathML"`,

`author_definition (x) <== "expert_SVG"`

It means that `author_definition (x)` has multiple assigned values.

The conditions of GAIE rules have an implicit "and". Similarly the actions.

**C) An Example of GAIE Rule**

```

Startrule "Related fragments"
  If  author(fragment_1) = x /* "x" is an author of fragment_1 */
      role(x) = "Writer" /* "x" has a writer role on fragment_1 */
      nature(fragment_1) = "image" /* fragment_1 is an image */
      action(x) = "modify_svg_element" /* x modifies the image */
      update(fragment_1) = "true" /* fragment_1 is saved */
      author(fragment_2) = y /* "y" is an author of fragment_2 */
      role(y) = "Writer" /* "y" has a writer role on fragment_2 */
      nature(fragment_2) = "text" /* fragment_2 is a text */
      reference(fragment_2) = fragment_1
      /* fragment_2 makes reference to the fragment_1 */
  Then
    announce(y) <- "referred object is modified"
    /* The modification of fragment_1 is notified */
Endrule

```

# Appendix B

## The GAIE Rule Base

### The Context-based Communication

```
Startrule "Work focus communication"
  If author(fragment_1) = x
    author(fragment_1) = y
    role(x) = "Writer"
    role(y) = "Reader" /* y can read fragment_1 */
    sync_comm(x) = "true" /* communication is enabled */
    sync_comm(y) = "true"
    action(y) = "select_element" /* action of y is selection */
  Then
    diplay(y) <- send(id)
    /* Unique identifier of the selected section is recovered and transmitted */
    announce(y) <- "Information is successfully sent"

Endrule
```

```
Startrule "Perception of a coauthor"
  If author(fragment_1) = x
    session(x) = "true"
    sync_comm(x) = "true"
  Then
    announce(x) <- "Information received for the work focus"
    display(x) <- perceive(selected_element)
    /* x perceives the work focus */

Endrule
```

### Work Proximity

```
Startrule "Related fragments"
  If author(fragment_1) = x
    role(x) = "Writer"
```

```

nature(fragment_1) = "image" /* fragment_1 is an image */
update(fragment_1) = "true" /* fragment_1 is saved */
author(fragment_2) = y
role(y) = "Writer"
nature(fragment_2) = "text" /* fragment_2 is a text */
reference(fragment_2) = fragment_1
    /* fragment_2 makes reference to the fragment_1 */
Then
    announce(y) <- "referred object is modified"
Endrule

```

## Notification

```

Startrule "A reviser opens a shared document"
If author(fragment_1) = x
    role(x) = "Writer"
    session_on(x) = "true"
    author(fragment_1) = y
    role(y) = "annotator"
    annotation(fragment_1) = "true"
        /* the author "y" can read and annotate the fragment_1 */
    action (y) = "open_document"
Then
    announce(x) <- "annotator opens document"
        /* author "x" is informed that author "y" starts the session */
Endrule

```

```

Startrule "Release of a shared document"
If author(fragment_1) = x
    role(x) = "Writer"
    session_on(x) = "true"
    annotation(fragment_1) = "true"
    role(y) = "annotator"
    action(y) = "close_document"
Then
    announce(x) <- "annotator closes shared document"
        /*coauthor "x" is informed that coauthor "y" closes the document */
Endrule

```

```

Startrule "Recording idle time and production"
If author(fragment_1) = x
    role(x) = "Writer"
    start_session(x) = "true"

```

```

Then
  initial_time(x) <- current_time(x)
  /* the current time is recorded */
  session_on(x) <- "true"
  /* to record that session is on */
  meta_data(fragment_1) <- summary(document)
  /* the contents of the document are registered */

Endrule

Startrule "Calculating idle time"
  If author(fragment_1) = x
    role(x) = "Writer"
    session_on(x) = "true"
  Then
    idle_time(x) <- current_time(x) - initial_time(x)
    /* the idle time is calculated */

  production(fragment_1) <- statistic(fragment_1)
  /* the contents of the document are registered */

Endrule

Startrule "Inactiveness"
  If author(fragment_1) = x
    role(x) = "Writer"
    session_on(x) = "true"
    idle_time(x) > 10 /* time in minutes */
    author(fragment_1) = y
    role(y) = "Manager"
    production(fragment_1) = Null
    communication_interest(y) = x
    /* the coauthor "y" wants to communicate with "x" */
  Then
    announce(y) <- "production of coauthor 'x' is null"
    /* the 'x' is inactive */

Endrule

Startrule "Notification service"
  If author(fragment_1) = x
    role(x) = "Writer"
    session_on(x) = "true"
    notification(x) = "disable" /* x disables notifications */
    author(fragment_2) = y

```

```

    role(y) = "Writer"
    session_on(y) = "true"
    communication_interest(y) = x
  Then
    announce(y) <- "The notification service is disabled"
Endrule

```

```

Startrule "Summary of a document"
  If action(x) = "open_document"
  Then
    meta_data(document) <- summary(document)
    /* The elements added or deleted from the document */
    announce(x) <- meta_data(document)
    /* Number of formulas, links, images, paragraphs */
Endrule

```

## Assistance

```

Startrule "Selecting MathML object"
  If author(fragment_1) = x
    nature(fragment_1) = "formula"
    action(x) = "select_element"
  Then
    announce(x) <- evaluate_wff(fragment_1)
    /* The coauthor is informed if a formula is well-formed */
    announce(x) <- list_tools(math_handling)
    /* functions for formulas */
Endrule

```

Endrule

```

Startrule "Well-formed formula"
  If author(fragment_1) = x
    role(x) = "Writer"
    nature(fragment_1) = "formula"
    update(fragment_1) = "true" /* fragment_1 is saved */
  Then
    announce (x) <- evaluate_wff(fragment_1)
    /* Well-formed formulas are announced to 'x' */
Endrule

```

Endrule

```

Startrule "Presence of MathML expert"
  If author(fragment_1) = x
    role(x) = "Writer"
    status_open(authors_definition_data_base) = "true"
Endrule

```

```

        /* experts in producing MathML elements exist in the database */
        expertise(x) <> "MathML"
        /* 'x' has not expertise to produce MathML components */
        action(x) = "start_formula"
    Then
        announce(x) <- "MathML_expert_exists"

Endrule

Startrule "Presence of MathML expert"
    If author(fragment_1) = x
        role(x) = "Writer"
        status_open(authors_definition_data_base) = "true"
        /* experts in producing MathML elements exist in the database */
        expertise(x) <> "MathML"
        /* 'x' has not expertise to produce MathML components*/
        action(x) = "start_formula"
    Then
        announce(x) <- "MathML_expert_exists"

Endrule

Startrule "SVG object handling"
    If author(fragment_1) = x
        nature(fragment_1) = "image"
        action(x) = "select_element"
    Then
        announce(x) <- list_tools(SVG_handling)
        /* SVG handling commands */

Endrule

Startrule "Table handling"
    If author(fragment_1) = x
        nature(fragment_1) = "table"
        action(x) = "select_element"
    Then
        announce(x) <- list_tools(table_handling)
        /* Table handling commands */

Endrule

```

## Evaluation of the Cooperative Production

```

Startrule "Counting the well-formed formulas"
  If author(fragment_1) = x
    role(x) = "Writer"
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    evaluate_wff(fragment_1) = "true" /* formula is wff */
  Then
    production(x) <- summary_wff(fragment_1)
    /* Well-formed formulas are counted and announced to 'x' */

```

```

Endrule

```

```

Startrule "Computing complexity"
  If author(fragment_1) = x
    role(x) = "Writer"
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    evaluate_wff(fragment_1) = "true" /* formula is wff */
  Then
    announce(x) <- computed_complexity(fragment_1)
    /* The complexity of the formula is calculated */

```

```

Endrule

```

```

Startrule "Computing complexity on request"
  If action(x) = "select_formula"
  Then
    announce <- computed_complexity(fragment_1)
    /* The complexity of the formula is calculated */

```

```

Endrule

```

```

Startrule "Actions-time-size"
  If action(x) = "start_formula"
    nature(fragment_1) = "formula"
  Then
    total_actions(x) <- start_count_total_actions(x)
    /* actions are counted */
    undo_actions(x) <- start_count_undo_actions(x)
    /* reverted actions are counted */
    initial_time(fragment_1) <- register_current_time(fragment_1)
    /* the time is recorded */\end{flushright}
    initial_size(document) <- register_initial_size(document)

```



```
/* the size is recorded */
```

```
Endrule
```

```
Startrule "Reverted actions"
```

```
  If author(fragment_1) = x
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    undo_actions(x) > total_actions / 2
      /* author reverts more than half of performed actions */
    status_open(authors_definition_data_base) = "true"
  Then
    announce(x) <- "MathML_expert_exists"
```

```
Endrule
```

```
Startrule "Counting the well-formed formulas"
```

```
  If nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    evaluate_wff(fragment_1) = "true" /* formula is wff */
  Then
    summary_wff(x) <- summary_wff(x) + 1
    /* Well-formed formulas are counted */
```

```
Endrule
```

```
Startrule "Final time-Final size"
```

```
  If author(fragment_1) = x
    role(x) = "Writer"
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
  Then
    final_time(x) <- register_current_time(x)
      /* the time is recorded */
    final_size(fragment_1) <- register_final_size(fragment_1)
      /* the size is recorded */
    elapsed_time(x) <- final_time(x) - initial_time(x)
      /* the elapsed time is computed */
    caculate_change_size(fragment_1) <-
      final_size(fragment_1) - initial_size(fragment_1)
      /* the change in size is computed */
```

```
Endrule
```

```
Startrule "Large elapsed time"
```

```
  If author(fragment_1) = x
```

```

nature(fragment_1) = "formula"
update(fragment_1) = "true"
elapsed_time(x) > 10      /* time in minutes */
complexity(fragment_1) < 16
status_open(authors_definition_data_base) = "true"
Then
  announce(x) <- "MathML_expert_exists"

```

Endrule

```

Startrule "Change in size"
  If author(fragment_1) = x
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    caculate_change_size(fragment_1) < 10 /* size in bytes */
    elapsed_time(x) > 1      /* time in minutes */
    status_open(authors_definition_data_base) = "true"
  Then
    announce(x) <- "MathML_expert_exists"

```

Endrule

```

Startrule "Defining expertise on writing formulas"
  If author(fragment_1) = x
    role(x) = "Writer"
    nature(fragment_1) = "formula"
    update(fragment_1) = "true"
    computed_complexity(fragment_1) > 16 /* from table 1 */
    summary_wff(formula) > 5      /* formulas are counted */
    status_open(authors_definition_data_base) = "true"
    /* author's definition file is open */
  Then
    author_definition(x) <== "expert_MathML"
    /*'x' is defined as an expert in writing mathematical formula */

```

Endrule

## MathML Transformations

```

Startrule "Exchanging patterns ((A,1) (B,1))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 1
  Then
    FirstChildNode(B) <- FirstChildNode(A)

```

Endrule

```
Startrule "Completing the first component ((A, 1) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 2
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    Blinking (SecondChildNode(B))
      /* Blinking the second empty node */
```

Endrule

```
Startrule "Completing the second component ((A, 1) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 2
  Then
    SecondChildNode(B) <- FirstChildNode(A)
    Blinking (FirstChildNode(B))
      /* the first node is empty */
```

Endrule

```
Startrule "Two remaining vacant components ((A, 1) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 3
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    Blinking (SecondChildNode(B))
      /* the second node is blinking */
    Blinking (ThirdChildNode(B))
      /* the third node is blinking */
```

Endrule

```
Startrule "First and third vacant components ((A, 1) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 3
```

```

Then
  SecondChildNode(B) <- FirstChildNode(A)
  Blinking (FirstChildNode(B))
      /* the first node is blinking */
  Blinking(ThirdChildNode(B))
      /* the third node is blinking */

Endrule

Startrule "First and second vacant components ((A, 1) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 1
    Number_parameter(B) = 3
  Then
    ThirdChildNode(B) <- FirstChildNode(A)
    Blinking (SecondChildNode(B))
      /* the second node is blinking */
    Blinking(FirstChildNode(B))
      /* the third node is blinking */

Endrule

Startrule "Exchanging contents ((A, 2) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 2
    Number_parameter(B) = 2
    RootNode(A) = RootNode(B)
      /* Both patterns are the same */
  Then
    FirstChildNode(B) <- SecondChildNode(A)
      /* exchange contents */
    SecondChildNode(B) <- FirstChildNode(A)

Endrule

Startrule "Exchanging operators ((A, 2) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 2
    Number_parameter(B) = 2
    RootNode(A) <> RootNode(B)
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- SecondChildNode(A)

```

Endrule

```
Startrule "From two to three components ((A, 2) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 2
    Number_parameter(B) = 3
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- SecondChildNode(A)
    /* the first and second node are completed */
    Blinking(ThirdChildNode(B))
    /* the third node is vacant */\end{flushright}
```

Endrule

```
Startrule "From two to three components and second vacant ((A, 2) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 2
    Number_parameter(B) = 3
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    ThirdChildNode(B) <- SecondChildNode(A)
    Blinking(SecondChildNode(B))
    /* the second node is vacant */
```

Endrule

```
Startrule "Reducing components ((A, 2) (B, 1))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 2
    Number_parameter(B) = 1
  Then
    FirstChildNode(B) <- FirstChildNode(A)
```

Endrule

```
Startrule "Reducing components ((A, 2) (B, 1))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 2
    Number_parameter(B) = 1
```

```

Then
  FirstChildNode(B) <- SecondChildNode(A)

Endrule

Startrule "First exchanging contents ((A, 3) (B, 3), (A=B))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 3
    RootNode(A) = RootNode(B)
  Then
    FirstChildNode(B) <- SecondChildNode(A)
    SecondChildNode(B) <- FirstChildNode(A)
    ThirdChildNode(B) <- ThirdChildNode(A)
    /* the first combination */

Endrule

Startrule "Second exchanging contents ((A, 3) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 3
    RootNode(A) = RootNode(B)
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- ThirdChildNode(A)
    ThirdChildNode(B) <- SecondChildNode(A)
    /* the second combination */

Endrule

Startrule "Second exchanging contents ((A, 3) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 3
    RootNode(A) = RootNode(B)
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- ThirdChildNode(A)
    ThirdChildNode(B) <- SecondChildNode(A)
    /* the second combination */

Endrule

```

```

Startrule "Third exchanging contents ((A, 3) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 3
    RootNode(A) = RootNode(B)
  Then
    FirstChildNode(B) <- ThirdChildNode(A)
    SecondChildNode(B) <- FirstChildNode(A)
    ThirdChildNode(B) <- SecondChildNode(A)
    /* the third combination */

```

Endrule

```

Startrule "Rename pattern ((A, 3) (B, 3))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 3
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- SecondChildNode(A)
    ThirdChildNode(B) <- ThirdChildNode(A)
    /* Each component is copied in the same order */

```

Endrule

```

Startrule "First reduction from 3 to 2 components ((A, 3) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 2
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- SecondChildNode(A)
    /* First possibility */

```

Endrule

```

Startrule "Second reduction from 3 to 2 components ((A, 3) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 2

```

```

Then
  FirstChildNode(B) <- SecondChildNode(A)
  SecondChildNode(B) <- ThirdChildNode(A)
                        /* Second possibility */

Endrule

Startrule "Third reduction from 3 to 2 components ((A, 3) (B, 2))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 2
  Then
    FirstChildNode(B) <- FirstChildNode(A)
    SecondChildNode(B) <- ThirdChildNode(A)
                        /* Third possibility */

Endrule

Startrule "First reduction from 3 to 1 component ((A, 3) (B, 1))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 1
  Then
    FirstChildNode(B) <- FirstChildNode(A)
                        /* The first possibility */

Endrule}

Startrule "Second reduction from 3 to 1 component ((A, 3) (B, 1))"
  If nature(fragment_1) = "formula"
    action(x) = "transform_element"
    Number_parameter(A) = 3
    Number_parameter(B) = 1
  Then
    FirstChildNode(B) <- SecondChildNode(A)
                        /* The second possibility */

Endrule

Startrule "Third reduction from 3 to 1 component ((A, 3) (B, 1))"

```



```
If nature(fragment_1) = "formula"
  action(x) = "transform_element"
  Number_parameter(A) = 3
  Number_parameter(B) = 1
Then
  FirstChildNode(B) <- ThirdChildNode(A)
  /* The third possibility */

Endrule
```



# Appendix C

## GAIE Functions

**Functions with a Single Value (1:1).** **author** (fragment) = x

Defines the set x of coauthors and their roles on the document fragment.

**action** (x) = "action\_name"

The coauthor x is performing writing action "action\_name".

**role** (x) = "role\_value"

The set x of coauthors having the role "role\_value".

**session\_on** (x) = "true"

The coauthor x has the session open.

**start\_session** (x) = "true"

The coauthor x starts the cooperative session.

**sync\_comm** (x) = "true"

The coauthor x has the synchronous communication available.

**nature** (fragment) = "image"

The nature of the fragment is "image".

**update** (fragment) = "true"

The document fragment is updated.

**notification** (x) = "disable"

The coauthor has disabled the notification functionalities.

**annotation** (fragment) = "true"

The fragment is annotated.

**shared\_obejct\_DB** (formula) = "true"

A MathML formula exists in the shared object database.

**evaluate\_wff** (formula) = "true"

The produced formula is well-formed .

**status\_open** (author\_definition\_data\_base) = "true"

The author definition database is open.

**Functions with Multiple Values (1:N).** **communication\_interest** (x) = y

The co-author x is interested to communicate with the co-author y.

**reference** (fragment\_2) = fragment\_1

The fragment\_2 is related to the fragment\_1.

**expertise** (x) = "MathML"

The co-author x is expert in writing MathML formulas.

**Numeric Functions.** **production** (fragment) = 10

10 elements have been produced on the fragment.

**idle\_time** (x) > 10

The co-author x is inactive for more than 10 minutes.

**computed\_complexity** (fragment) > 16

The associated complexity of the fragment (formula) is greater than 16.

**summary\_wff** (formula) > 5

There are 5 well-formed formulas in a fragment.

**undo\_actions** (x) > 10

The co-author x has reverted 10 actions.

**elapsed\_time** (x) > 15

The elapsed time of the co-author x is more than 15 minutes.

**calculate\_change\_size** (fragment) < 10

The change in size of the fragment is less than 10 bytes.

# Appendix D

## GAIE Actions

1. **announce** (x) - It notifies a user of different activities of the coauthors in the cooperative authoring process. The user who is intended for notification becomes the argument of this action. A string of characters becomes the message for notification send to a user.

`announce(x) <- <notification message>`

2. **display** (x) - It displays the "work focus" in the coauthor's environment.

`display (x) <- send (id)`

3. **idle\_time** - It calculates the difference of two times when the user is not performing any action in the cooperative session. It takes the integer value in minutes.

`idle_time (x) <- <integer>`

4. **initial\_time** (x) - It records the time when a user starts to produce a document section (a formula, a figure, etc.). It takes the value in time formate.

`initial_time (x) <- system_time(x)`

5. **session\_on** (x) - It registers that the user is producing, i.e. a user is active in the session. It take the "true" or "false" value.

`session_on <- "true"`

6. **meta\_data** (document) - It gives the information about components composing a shared document, e.g number of total elements, percentage of elements of MathML, SVG, and XHTML schemas, well-formed expressions, table, etc.

`meta_data (document) <- summary (document)`

7. **total\_actions** (x) - It registers the number of total actions performed by a user, from the start of production of a fragment to its termination. It takes an integer value.

`total_actions (x) <- start_count_total_actions (x)`

8. **undo\_action** (x) - It registers the number of total undo actions performed by a user, from the start of production of a fragment to its termination. It takes an integer value.  
`total_actions (x) <- start_count_undo_actions (x)`
9. **final\_time** (x) - It records the time when a user terminate a document section (a formula, a figure, etc.). It takes the value in time formate.  
`final_time (x) <- system_time(x)`
10. **initial\_size** (document) - It registers the size of the document when a user starts to produce a document section (a formula, a figure, etc.). It takes an integer value.  
`initial_size(document) <- register_initial_size(document)`
11. **final\_size** (document) - It registers the size of the document when a user terminate a document section (a formula, a figure, etc.). It takes an integer value.  
`final_size(document) <- register_final_size(document)`
12. **elapsed\_time** (x) - It calculates the difference between registered final and initial times when the user respectively terminates and starts a document section. It takes an integer value.  
`elapsed_time (x) <- final_time (x) - initial_time (x)`
13. **calculate\_change\_size** (fragment\_1) - It calculates the difference between registered final and initial sizes of the document when the user respectively terminates and start a document section. It takes an integer value.  
`calculate_change_size (fragment_1) <- final_size (fragment_1) - initial_size (fragment_1)`
14. **statistic** (document) - It compares the contents of the shared document between a specified interval, e.g. when a document is opened and when it is updated
15. **production** - This action informs a user the number of added, deleted or modified element in a fragment.
16. **summary** - This actions informs a summary of the elements contained in the shared document. The elements are formulas, tables, figures, etc.
17. **list\_tools** - This action displays a list of available facilities for handling a particular object like "formula", "SVG\_figure".
18. **evaluate\_wff** - This action evaluate a MathML expression to be well-formed and then, it is informed to a coauthor.
19. **summary\_wff** - This action informs a coauthor the number of well-formed formulas in the shared document.

20. **calculate\_complexity** - This action calculates the associated complexity of a formula, and informs this complexity to the coauthor.
21. **author\_definition** - This action defines a coauthor in the author's definition file as an expert in "MathML", "SVG\_drawing", and "Structured\_text"  
author\_definition (x) <- <string of characters>





# Appendix E

## Glossary

In this appendix, we define the words that are written in *italic* throughout the thesis. We group these words under the CSCW domain, the AllianceWeb/ PINAS project, and the GAIE inference engine.

### **Computer Supported Cooperative Work (CSCW)**

**Awareness** is an understanding of the activities of others, which provides a context of your own activity.

**Communication** is a discipline in charge of conveying information or ideas by speaking, signaling or writing, between people or groups.

**Computer Supported Cooperative Word (CSCW)** is a generic term which combines the understanding of the way people work in groups with the enabling technologies of computer networking, and associated hardware, software, services and techniques.

**Coordination** is the support for the activity of managing dependencies and possible conflicts between collaborative entities (users and their roles) involved in common and inter-related tasks of a collaborative activity (actions performed in the shared workspace).

**Cooperative activities** are performed to achieve a common goal, e.g. the production of a shared document.

**Group awareness** is the understanding of who is working with you, what they are doing and how your own actions interact with their ones.

**Groupware** is the computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment. A groupware system can involve software, hardware, services, and group process support. The cooperative writing applications like AllianceWeb, BSCW, EquiText, are examples of a groupware system.

**PIÑAS/AllianceWeb Project**

**AllianceWeb user** is a physical person that can perform authoring actions on the shared document.

**An authoring action** is an action performed by a user during the cooperative production, e.g. opening and closing of a document, annotation, edition, deletion, selection.

**Cooperative authoring environment** allows coauthors to directly access and cooperatively modify documents inside the Web publishing environment.

**Distributed Event Management Service (DEMS)** is responsible to capture, manage, filter events from event producer applications, and to deliver events to event consumer applications.

**Editing roles** is the authorization of a user to edit a fragment: [**M-Manager**, **W-Writer**, **R-Reader**, **N-Null**].

**Event** is an authoring action represented, inside the DEMS service, e.g. starting a session, opening a document, selecting a formula.

**Event consumer application** consume events, e.g. awareness functionalities.

**Event producer application** generate events, e.g. AllianceWeb cooperative writing editor.

**Fragmentation** is the splitting of the shared document into sharing units (fragments) depending on its logical structure. For instance, the fragments of the shared document of type article: title, abstract, sections, paragraphs, bibliography.

**Fragment nature** is the nature of produced fragment, e.g. a MathML expression, a SVG graphic, a XHTML text.

**Manager role** can assign the writer, the reader, and the null roles to coauthors, on the fragments of the shared document.

**Null role** is the right to do not edit and do not read a document fragment.

**Reader role** is the right to read a document fragment.

**Resources** are the entities that are shared by one or several documents, e.g. roles, formulas, figures, tables, sections, paragraphs.

**Storage site** is the site on which stable versions of the documents and their resources are saved.

**Working site** is the site from where a coauthor starts a cooperative session and performs authoring actions.

**Writer role** is the right to edit a document fragment. The writer role is exclusively controlled, i.e. only one author can write on a fragment at one time.

### **Group Awareness Inference Engine (GAIE)**

**Announce** is the information provided to users by the GAIE inference engine depending on the handled objects and performed authoring actions.

**Assistance** is the information provided by the GAIE inference engine about the presence of experts or implicit functions (keyboard shortcuts) of the authoring application.

**Beginner** is a user who spends a lot of time and frequently reverts his performed actions when he produces a Math expression in our authoring application.

**Complexity of a MathML formula** is a real number value given to a MathML formula, considering the steps taken to produce it like selecting a menu, MathML dialog box, math palette, etc.

**Context-based communication** is a quasi synchronous exchange of messages between two coauthors over a produced object (a formula, a table, a figure, a text) within the shared document.

**MathML expert** is a user who produces elaborated well-formed MathML expressions using MathML structures in our authoring application.

**Unique identifier** is an identifier associated with each component of the shared document.

**Work proximity.** When at least two objects are perceived as belonging to a unit, we say that these objects are near/close. We denominate this property as work proximity.

**Transformation of MathML expressions** is the process to get a MathML expression from the given one reusing its contents.

**MathML Well-formed formula (wff)** is an expression whose all components are completed and each component, in turn is a well-formed formula too.

**Work focus** is the object on which the coauthors center their discussion.



# Bibliography

- [1] Applet, W., “WWW Based Collaboration with the BSCW System”, *Proc. of the 24th Seminar on Current Trends in Theory and Practice of Informatics (SOFSEM'99)*, Springer Lectures Notes in Computer Sciences, vol. 1975, Milovy, pp. 66-78, December 1999.
- [2] Bannon, L., Schmidt, K., “CSCW: Four Characteristics in Search of a Context”, *Proc. of the first European Conference on Computer Supported Cooperative Work (ECSCW'89)*, Gatwick, London, vol. 13-14 September, 1989, pp. 358-372. - Reprinted in *Studies in Computer Supported Cooperative Work. Theory, Practice and Design*, J. M. Bowers and S. D. Benford, Eds. North-Holland, Amsterdam, pp. 3-16, 1991.
- [3] Bentley, R., Horstmann, T., Trevor, J., “The World Wide Web as Enabling Technology for CSCW: The Case Study of BSCW. Computer Supported Cooperative Work”, *The Journal of Collaborative Computing*, vol. 6, num. 2-3, pp. 111-134, 1997.
- [4] Bentley, R., Applet, W., Busbach, U. Hinrichs, E., Kerr, D., Sikkel, K., Trevor, J., Woetzel, G., “Basic Support for Cooperative Work on the World Wide Web”, *International Journal of Human Computer Studies: Special Issue on Novel Applications of the WWW*, vol. 46, num. 6, pp. 827-846, 1997.
- [5] Bradley, N., *The XML Companion, Third edition*, Addison-Wesley Pub. Co., 2002.
- [6] Budzik, J., Fu, X., Hammond, K. J., “Facilitating Opportunistic Communication by Tracking the Documents People Use”, *ACM CSCW' 2000 Workshop on Awareness and the WWW*, 2000.
- [7] Cadiz, J. J., Danielle, G. V., Jancke, G., Gupta, A, “Designing and Deploying an Information Awareness Interface”, *Proc. on the ACM 2002 Conference on Computer Supported Cooperative Work New Orleans, Louisiana, U.S.A.*, pp. 314-323, November 16-20, 2002.
- [8] Carlisle, C., Ion, P., Miner, R., Poppelier, N., “Mathematical Markup Language (MathML) Version 2.0 (Second Edition)”, *W3C Recommendation*, October 21, 2003. Available at <http://www.w3.org/TR/MathML2>.

- [9] Carroll, J. M., Neale, D. C. Isenhour, P. L., "Notification and Awareness: Synchronizing Task-oriented Collaborative Activity", *International Journal of Human-Computer Studies*, vol. 58, May 2003.
- [10] Cohen, D., Jacovi, M., Marrek, Y., Soroka, V., "Collection Awareness on the Web via Livemaps", *ACM, SIGGROUP Bulletin*, vol. 21, issue 3, pp. 12-15, December 2000.
- [11] Decouchant, D., Jesús, F., Martínez Enríquez, A. M., "PIÑAS: A Middleware for Web Distributed Cooperative Authoring", *Proc. of the SAINT'2001, San Diego, CA (U.S.A.)*, pp. 187-194, January 2001.
- [12] Decouchant, D., Martínez Enríquez, A. M., Jesús, F., Morán, A., Mendoza, S., Jafar, S., "A Distributed Event Service for Adaptive Group Awareness", *Proc. of the Second Mexican International Conference on Artificial Intelligence, Mérida, Yucatán, Mexico, Lecture Notes in Artificial Intelligence (LNAI-2313), Springer-Verlag*, pp. 506-515, April 2002.
- [13] Decouchant, D., Quint, V., Salcedo M. R., "Structured and Distributed Cooperative Editing in a Large Scale Network", *Groupware and Authoring (Chapter 13), R. Rada, Academic Press London, UK*, pp. 265-295, May 1996.
- [14] Decouchant, D., Martínez Enríquez, A. M., Martínez, E., "Document for Web Cooperative Authoring", *Proc. of the CRIWG'99, the 5th International Workshop on Groupware, IEEE Computer Society, Cancún (Mexico)*, pp. 15-18, September 1999.
- [15] Ellis, C. A., Gibbs, S. J., Rein, G. L., "Groupware: Some Issues and Experiences", *Communication of the ACM*, vol. 34, num. 1, pp. 38-58, January 1991.
- [16] Erickson, T., Halverson, C., Kellogg, W. A., Laff, M., Wolf, T., "Social Translucence: Designing Social Infrastructures That Make Collective Activity Visible", *Communications of the ACM (CACM)*, vol. 45, num. 4, pp. 40-44, April 2002.
- [17] Feigenbaum, E. A., McCorduck, P., "*The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*", Addison-Wesley, 1983.
- [18] Fernández, L., Sánchez, J. A., Flores, A., "An Environment for the Collaborative Revision of Digital Theses", *Proc. of the 6th International Workshop on Groupware (CRIWG 2000), Madeira, Portugal. IEEE Computer Society*, pp. 150-153, October 18-20, 2000.
- [19] Flores, F., Graves, M., Hartfield, B., Winograd, T., "Computer Systems and the Design of Organizational Interaction", *ACM Transaction on Office Information Systems*, vol. 6, num. 2, pp. 153-172, April 1988.
- [20] Fuchs, L. U. P., Wolfgang Prinz, "Supporting Cooperative Awareness with Local Event Mechanisms: The GroupDesk System", *Proc. of ECSCW'95*, 1995.

- [21] Goderfoid, P., Herbsleb, J. D., Jagadeesan, L. J., Li, D., "Ensuring Privacy in Presence Awareness Systems: An Automated Verification Approach", *Proc. of ACM CSCW'00 (Philadelphia, PE, U.S.A.)*, ACM Press, pp. 59-68, December 2000.
- [22] Golang, Y. Y., Whitehead, E. J., Faisy, A. Jr., Carter, S. R., Jensen, D., "HTTP Extension for Distributed Authoring - WebDav", *RFC*, vol. 2518, February 1999.
- [23] Grudin, J., "Groupware and Social Dynamics: Eight Challenges for Developers.", *Communication of the ACM (CACM)*, vol. 37, no. 1, pp. 92-105, January 1994.
- [24] Guetari, R., Quint, V., Vatton, I., "Amaya: an Authoring Tool for the Web", *Proc. of MCSEAI'98 Maghrebian Conference on Software Engineering and Artificial Intelligence, Tunis, Tunisia*, 8-10 December 1998 (<http://www.w3.org/Amaya>).
- [25] Grudin, J., "Computer Supported Cooperative Work: History and Focus", *IEEE Computer*, vol. 7, no. 5, pp. 19-26, 1994.
- [26] Gutwin, C., Greenberg, S., "A Descriptive Framework of Workspace Awareness for Real-Time Groupware", *Computer Supported Cooperative Work (CSCW)*, vol. 11, num. 3-4, pp. 411-446, 2002.
- [27] Gutwin, C., Penner, R., Schneider, K., "Group Awareness in Distributed Software Development", *Proc. of the 2004 ACM conference on Computer Supported Cooperative Work 2004, Chicago, Illinois, U.S.A.*, pp. 72-81, November 06-10, 2004.
- [28] Gutwin, C., Greenberg, S., Roseman, M., "Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation", *Proc. of the HCI'96*, pp. 281-298, August 20-23, 1996.
- [29] Gutwin, C., Stark, G., Greenberg, S., "Support for Workspace Awareness in Educational Groupware", *Proc. of ACM Conference on Computer Supported Collaborative Learning*, Indiana, U.S.A., pp. 147-156, October 17-20, 1995.
- [30] Guzdial, M., Rick, J., Kerimbaev B, "Recognizing and Supporting Roles in CSCW", *Proc. of the 2000 ACM Conference on Computer Supported Cooperative Work, Philadelphia, Pennsylvania, U.S.A.*, ISBN: 1-58113-222-0, pp. 261-268, 2000.
- [31] Henrique J. L. D., Nuno M. P., Legatheaux, J. M., "Coordination and Awareness Support for Adaptive CSCW Sessions", *Proc. of the 4th CYTED\_RITOS International Workshop in Groupware Systems, Brazil*, 1998.
- [32] Hiltz, S. R., Turoff, M., *The Network Nation*, Revised Edition, Cambridge, MA; MIT Press, 1993, Original Edition 1978.
- [33] Hudson, S. E., Smith, I. E., "Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems", *Proc. of the ACM 1996 Conference on Computer Supported Cooperative Work, MA, U.S.A.*, pp. 248-257, November 16-20, 1996.

- [34] Hughes, G., Crowder, R., "Experiences in Designing Highly Adaptable Expertise Finder Systems", *Proc. of DETC'03- ASME 2003 Design Engineering Technical Conferences and Computer and Information in Engineering Conference, Chicago, Illinois, U.S.A.*, September 2-6, 2003.
- [35] Ishikawa, M., McCarron, S., Pemberton, S., "HTML 2.0", *W3C Working Draft. Edited by Jonny Axelsson (Opera Software)*, May 6, 2003. Also available at <http://www.w3.org/TR/xhtml2>.
- [36] Jang, C., Steinfield, C., Pfaff, B., " Supporting Awareness Among Virtual Teams in a Web-Based Collaborative System: The TeamSCOPE system", *International Workshop on Awareness and the World Wide Web, CSCW 2000, Philadelphia, PA. U.S.A.*, December 2000.
- [37] Jensen, C., Farnham, S. D., Drucker, S. M., Kollock, P., "The Effect of Communication Modality on Cooperation in Online Environment", *Proc. of the Conference on Human Computer Factors in Computing Systems, CHI'2000, The Hague, The Netherlands, ACM, 2000*, pp. 470-477, April 1-6, 2000.
- [38] Johansen, R., *Groupware: Computer Support for Business Teams*, New York: Free Press, U.S.A, 1988.
- [39] Jokinen, K., Kanto, K., "User Expertise Modelling and Adaptivity in a Speech-Based E-mail System", *Proc. of the 42nd annual meeting of the Association of Computational Linguistics (ACL'2004), Barcelona, Spain*, pp. 87-94, July 21-26, 2004.
- [40] Judith, A. P., *New Technology, Surveillance and Social Control: Surveillance and Privacy in Computer Supported Cooperative Work*, University of Minnesota Press, 1995.
- [41] Kimura, H., Miyaji, C., "An Implementation of Internet Applications in Mathematica", *Proc. of the Second International Mathematica Symposium*, pp. 307-314, 1997 (<http://www.wolfram.com>).
- [42] Lukasiewicz, J., *Elements of Mathematical Logic*, Warsaw, 1st Edition, 1929.
- [43] Malone, T. W., Crowston, K., "The Interdisciplinary Study of Coordination", *ACM Computing Surveys*, vol. 26, num. 1, pp. 87-119, March 1994.
- [44] Manuel K., Lima, J. V., Borges M. R. C., "A Framework for Awareness Support in Groupware Systems", *Proc. of the 7th International Conference on CSCW in design - CSCWD'2002, Rio de Janeiro, Brazil*, pp. 13-18, September 2002.
- [45] Mark, G., Fuchs, L., Sohlenkamp, M., "Supporting Groupware Conventions Through Contextual Awareness", *Proc. of ECSCW'1997, Lancaster, England*, pp. 253-268, September 7-11, 1997.



- [46] McDaniel, S. E., Brinck, T., "Awareness in Collaborative Systems", *Proc. of the Conference on Human Factors in Computing Systems (Workshop) - CHI 97 Extended Abstracts, Atlanta, Georgia, U.S.A.*, pp. 237, March 22-27, 1997.
- [47] McDonald, D. W., Ackerman, M. S., "Just talk to me: A Field Study of Expertise Location", *Proc. of the 1998 ACM Computer Supported Cooperative Work (CSCW'98) Seattle, WA*, pp. 315-324, November 1998.
- [48] McDonald D. W., Ackerman M. S., "Expertise Recommender: A Flexible Recommendation System and Architecture", *Proc. of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW'00), Philadelphia. P.A.*, pp. 231-240, December 2000.
- [49] McDonald D. W., "Evaluating Expertise Recommendations", *Proc. of ACM 2001 International Conference on Supporting Group Work (GROUP'01)*, Boulder CO, October 2001.
- [50] Melis, E., Andrés, E., Búdenbender, J., Frischauf, A., Gogvadze, G., Libbrecht, P., Pollet, M., Ullrich, C., "ActiveMath: A Generic and Adaptive Web-Based Learning Environment", *International Journal of Artificial Intelligence in Education*, vol. 12, pp. 385-407, 2001 (<http://www.activemath.org>).
- [51] Muhammad, A., Martínez Enríquez, A. M., Decouchant, D., "Awareness and Coordination for Web Cooperative Authoring", *Proc. of 3rd Atlantic Web Intelligence Conference, AWIC 2005, Lodz, Poland*, pp. 327-333, June 6-9, 2005, Lecture Notes in Computer Science.
- [52] Noël, S., Robert, J. M., "Empirical Study on Collaborative Writing: What Do Co-authors Do, Use, and Like?", *Computer Supported Cooperative Work (CSCW)*, vol. 13, num. 1, pp. 63-89, 2004.
- [53] Ogata, H., Yano, Y., "Supporting Knowledge Awareness for a Ubiquitous", *CSCL, eLearn 2003, Phoenix, Arizona, U.S.A.*, pp. 2362-2369, November 7-11, 2003.
- [54] Paul, C. L., "Editing SVG with Other XML Languages", *Proc. of the SVG Open - First Conference on Scalable Vector Graphics, Zurich, Switzerland*, July 15-17, 2002.
- [55] Paul Dourish, Victoria Bellotti, "Awareness and Coordination in Shared Workspace", *CSCW 1992*, pp. 107 - 114, 1992.
- [56] Paul Dourish, Sara Bly, "Portholes: Supporting Awareness in a Distributed Work Group", *Proc. of Conference on Human Factors in Computing Systems (CHI'92)*, Monterey, CA, U.S.A., ACM, pp. 541-547, May 3-7, 1992.
- [57] Pedersen, E. R., "AROMA: Abstract Representation Of Presence Supporting Mutual Awareness", *Proc. of Conference on Human Factors in Computing Systems (CHI'97)*, Atlanta, Georgia, pp. 51-58, March 22-27, 1997.

- [58] Pinheiro, M. K., Lima J. V., “An Adaptable Framework for Past Events Awareness Support”, June 2001.
- [59] Quint, V., Vatton, I., “Making Structured Document Active”, *Electronic Publishing - Orientation, Dissemination and Design*, vol. 7, num. 1, pp. 55-74, March 1994.
- [60] Rizzi, C. B., Alonso, C. M. M. C., DE Seixas, L. M. J., Costa, J. S. Tamusiunas, F. R., DA Rosa Martins, A., “Collaborative Writing via Web - EquiText”, *Seventh International Conference on Informatic Education*, 2000.
- [61] Rizzi, C. B., Alonso, C. M. C., Hassan, E. B., Tarouco, L M. R., DE Seixas, L. M. J., “EquiText: A Helping Tool in the Elaboration of Collaborative Texts”, *Proc. of SITE'2000 - 11th International Conference*, 2000.
- [62] Sánchez, J. A., Flores, L. A., “Provisions for Collaborative Revision and Annotation of Digital Documents”, *Proc. of 2002 ACM Conference on Computer-Supported Cooperative Work, (ACM Press), Conference Supplements*, pp. 179-190, 2002.
- [63] Schmidt, K., Bannon, L., “Taking CSCW Seriously: Supporting Articulation Work”, *Computer Supported Cooperative Work (CSCW): An International Journal*, vol. 1, no. 1, pp. 7-40, June 1992.
- [64] Sison, R., Numao, M., Shamura M., “Multistrategy Discovery and Detection of Novice Errors”, *Machine Learning*, vol. 38, num. 1-2, pp. 157-180, 2000.
- [65] Sohlenkamp, M., Prinz, W., Fucha, L, “POLIAwac: Design and Evaluation of an Awareness Enhance Groupware Client”, *AI & Society and Journal*, vol. 14, pp. 31-47, 2000.
- [66] Smith, I. E., Hundson, S. E., Mynatt, E. D., Selbie, J. R., “Applying Cryptographic Techniques to Problems in Media Space Security”, *Proc. of ACM Conference On Organizational Computing Systems (COOCS'95), Milpitas, California, U.S.A.*, pp. 190-196, August 13-16, 1995.
- [67] Stefik, M., Bobrow, D. G., Foster, G., Lanning, S., Tatar, D. G., “WYSIWIS Revised: Early Experiences with Multiuser Interfaces”, *ACM Transaction on Office Information Systems (TOIS)*, vol. 5, no. 2, pp. 147-167, April 1987.
- [68] Suchman, L. A., Wynn, E., “Notes on Computer Support for Cooperative Work”, Dept. Computer Science, University of Jyvaskyla, SF-40100 Jyvaskyla, Finland, May 1989.
- [69] Tran, M. H., Raikundalia, G. K., Yang, Y., “Methodologies and Mechanism Design in Group Awareness Support for Internet-based Real-time Distributed Collaboration”, *Asia-Pacific Web Conference, Lecture Notes in Computer Science, Springer-Verlag*, vol. 2642, pp. 357-369, 2003.

- [70] Turoff, M., Hiltz, S. R., Bieber, M., Fjermestad, J., Rana, A., “Collaborative Discourse Structures in Computer Mediated Group Communications”, *Journal of Computer-Mediated Communication*, vol. 4, no. 4, June 1999.
- [71] Vertegaal, R., “Designing Awareness with Attention-based Groupware”, *Proc. of IN-TRACT’99, IFIP, Edinburg, UK*, 1999.
- [72] Yiman, D., Kobsa, A., “DEMOIR: A Hybrid Architecture for Expertise Modeling and Recommender Systems”, *IEEE 9th International Workshop on Enabling Technologies: Infrastructure for Collaborative Expertise (WET ICE’00)*, March 14-16, 2000.
- [73] Wilson, P., “*Computer Supported Cooperative Work*”, Oxford, UK: Intellect Books, 1991.
- [74] Wojtasiewicz, O., *Elements of Mathematical Logic*, The MacMillan Company, New York, 1963.
- [75] Yang, Y., Sun, C., Zhang, Y., Jia, X., “Real-time Cooperative Editing on the Internet”, *IEEE Internet Computing*, vol. 4, num. 3, pp. 18-25, May-June, 2000.
- [76] Yang, Y., “An Architecture and the Related Mechanisms for Web-based Global Cooperative Teamwork Support”, *International Journal of Computing and Informatics*, vol. 24, num. 1, pp. 13-29, 2000.