PROMISE

Participative Research labOratory for Multimedia and Multilingual Information Systems Evaluation

FP7 ICT 2009.4.3, Intelligent Information Management

# Deliverable 5.1

# Collaborative User Interface Requirements

Version 1.1, 28th February 2011

SEVENTH FRAMEWORK PROGRAMME

## Document Information

| | |
|---|---|
| **Deliverable number:** | D5.1 |
| **Deliverable title:** | Collaborative user interface requirements |
| **Delivery date:** | 28/02/2011 |
| **Lead contractor for this deliverable** | ROMA1 |
| **Author(s):** | Martina Croce, Emiliano Di Reto, Guido Lorenzo Granato, Preben Hansen, Alessandro Sabetta, Giuseppe Santucci, Fabio Veltri |
| **Participant(s):** | All partners |
| **Workpackage:** | WP5 |
| **Workpackage title:** | Collaboration and Knowledge sharing |
| **Workpackage leader:** | ROMA1 |
| **Dissemination Level:** | PU – Public |
| **Version:** | 1.1 - FINAL |
| **Keywords:** | Collaboration, knowledge sharing, user collaboration requirements |

## History of Versions

| Version | Date | Status | Author (Partner) | Description/Approval Level |
|---|---|---|---|---|
| 1.0 | 07/02/2011 | Draft | All | Circulated to all partners |
| 1.1 | 28/02/2011 | Final | All | Revised after partners' comments |

## Abstract

This deliverable reports the collaboration and knowledge sharing requirements of the PROMISE project. Collaboration is one of the core activities within PROMISE and one of the major goals of the PROMISE evaluation infrastructure which aims at improving the usage and the exploitation that developers, researchers, and stakeholders make of the scientific data produced by the experimental evaluation of multilingual and multimedia information access systems.

## Table of Contents

## Executive Summary

Work package 5 ("Collaboration and Knowledge Sharing") is responsible for designing, developing and delivering the user interfaces and the annotation service needed to promote the collaboration among the stakeholders of the evaluation infrastructure and foster the knowledge sharing and reuse. Moreover, it is responsible for exploring how to exploit information visualization and visual analytics techniques to information retrieval experimental data in order to improve their understanding and allow researchers to effectively cope with huge amount of data.

This deliverable, in particular, focuses on the collaboration aspects within PROMISE by discussing the problem of collaboration and knowledge sharing in general and in the information retrieval field, more specifically, and detailing the collaboration functionalities of PROMISE, namely:

- Annotations
- Polls
- Message forums

Then, the deliverable details the actual user requirements for collaborating and sharing knowledge by following the guidelines of the IEEE 830-1998 Recommended Practice Software Requirements Specifications standard.

Finally, the deliverable gives an outlook of what collaboration and knowledge sharing can mean from a visual analytics perspective. This will be further refined in deliverable D5.2 "User interface and Visual analytics environment requirements" due at month 12.

The appendices report all the needed background information.

# 1 Introduction

This deliverable reports the Promise collaborative user interface requirements, focusing on both knowledge sharing and collaboration issues. Requirements are described with UML Use Cases and using a textual template derived by the "IEEE standard 830-1998 - Recommended Practice for Software Requirements Specifications".

The deliverable is structured as follows. Section 2 describes the state of the art on collaboration and knowledge sharing in information retrieval and Section 3 and 4 present the requirements and the solutions selected for the Promise project.

Section 6 and Section 6 present the requirements following the IEEE standard 830-1998 and, together with Section 7 that summarizes the Promise requirements through UML Use Cases, can be used out of the context of this report.

Appendix 8.1 recall the evaluation infrastructure architecture defined in WP3 showing user classes and the associated requirements. Appendix 8.2 presents some related applications and projects dealing with collaboration issues. Appendix 8.3 describes the IEEE Recommended Practice for Software Requirements template. Appendix 8.4 contains a quick guide for UML Use Cases, and Appendix 5.5 presents a set of guidelines for user interface design and developments, based on the Nielsen's heuristics.

# 2 Collaboration and knowledge sharing in information retrieval

So far, there is no widely accepted definition of collaboration, sometimes it is also referred to as cooperation. As Foster (2006) correctly points out, research related to collaborative information seeking and retrieval is an interdisciplinary phenomenon including studies especially from areas such as human-computer interaction (HCI), computer-supported cooperative work (CSCW) and information science. Thus, definitions of collaborative information seeking are developed from the disciplines and circumstances they have been used in. In the present study, collaboration is specifically related to information retrieval (IR). Collaborative information retrieval means active and explicit retrieval of information for solving a specific task. Sharing information, on the other hand, is usually about sharing already acquired information. Sometime these do coincide (Hansen and Järvelin, 2005).

An early example is Allen (1977) who studied the differences between the information seeking behavior between engineers and scientists. Allen points out important aspects of the information seeking behavior relevant for our study: the importance of personal contacts and discussions between engineers and that there are gatekeepers in organizations. Allen also studied patterns of communication within a small research laboratory and found a typical communication network. These networks showed central points (persons) around which communication was centered.

Recent research in information seeking and retrieval (IS&R) extends our knowledge on how people access, retrieve and judge information.  Some examples are:

O'Day (1993) described four levels of sharing information in collaborative group situations: a) sharing results with other team members; b) self-initiated broadcasting of interesting information; c) handling search requests made by others; and d) archiving potentially useful information into group repositories for others to use.

Karamuftuoglu (1998) discusses what he calls social informatics, which seeks to include the relationships between humans within an IR process.

Fidel and colleagues (2000) describe a project focusing on collaborative activities of members of a work-team within an organization performing IS&R tasks.

Hansen and Järvelin (2000, 2005) investigated the IS&R processes performed by patent engineers. One of the main preliminary results in this study was that the patent engineers were involved in different collaborative activities.

Foster, J. (2006) presents a literature review describing current research of collaboration related to seeking, searching and retrieval tasks. The information-seeking task involves both social and collaborative approaches, while the information retrieval task involves collaborative issues such as collaborative filtering and collaborative querying. Foster concludes that research in the field of CIR needs to address and evaluate the conditions that influence development of systems that handle collaborative information activities, such as "…direct and indirect collaboration during information tasks…"

Reddy & Spence (2008) conducted an ethnographic field study of a multidisciplinary patient care team in an emergency department. The goal was to identify information needs within a medical team and to identify situations that trigger collaborative information seeking activities.

Empirical studies on collaboration in IR among end-users have been scarce indeed, but there is an increasing interest that results in contributions such as the SearchTogether system by Morris and Horwitz (2007), a prototype that allows a group of users to remotely collaborate when searching information on the Internet.

In HCI and CSCW we find a large body of literature where attempts are made to facilitate finding information through social networks, (e.g. The Answer Garden by Ackerman & Malone, 1990). Another attempt is made by McDonald & Ackerman, (1998) reporting on the Information Lens. They conducted a five-month field study on how people in a medium-sized organization find the expertise to construct, maintain and support their software systems.  The study deals mainly with how people share information through expertise identification and expertise selection. Research in Computer Supported Collaborative Work (CSCW) deals with collaboration in organizations and work groups, and systems supporting collaboration, such as organizational memory, organizational information handling and information sharing.

Traditional human communication may be asynchronous or synchronous - asynchronous through ordinary mail and book/journal reading; and synchronous through human face-to-face real-time communication and ad-hoc social interactions.  Computer-mediated communication may also be asynchronous through e.g. e-mail, searching the Internet and log viewing, and synchronous through video conferencing (e.g. Erlich & Cash, 1994; Haake et al., 1999); and b) Loosely or tightly coupled activities (Tang et al., 2006). In loosely coupled activities, the system will take advantage of recommendations from other people through observations of their information seeking behavior such as search paths and annotations; recommendations based on usage rates, and explicitly stated recommendations. Tightly coupled activities may in the context of IS&R include sharing queries and strategies for their refinement, and feedback and judgment phases with others (Haake et al., 1999).

# 3 Collaboration and knowledge sharing in Promise

Summarizing what has been presented in the previous section, collaborative software has the main goal of helping people involved in a common task to achieve their goals. The objective of this section is to describe what are the Promise collaborative activities that have to be supported by the system, and how. To fix the ideas, we provide a rough overview of the foreseen system functionalities, distinguishing two main levels:

- Knowledge-sharing, and
- Collaborative interaction

## 3.1 Promise functionalities supporting knowledge sharing

With the expression knowledge sharing we denote techniques that allow users to share information about their work (insights, useful data, interesting visualizations, etc.). We foresee to address this objective using a publish/subscribe mechanism together with an annotation system. This architectural pattern involves two kinds of users, publishers and subscribers, and allows subscribers for expressing their interest for particular events, in order to receive relevant notifications.

In particular, we foresee two publish/subscribe mechanisms:

- subject-based, and
- list-based

The subject-based publish/subscribe is a particular schema in which events (e.g., creating or annotating a document) are “tagged” with one or more identifiers (subjects) when are published. Subscribers issue subscriptions according to the subjects they are interested in. A subject can be represented as a “virtual channel” connecting producers to consumers. It is worth noting that, in a standard environment, when a user subscribes a subject, she will receive the information about that subject from that time on. That represents a limitation: the user might be also interested in all the work previously produced by the community. The Promise system should therefore give a user the option to receive all the past events associated with a particular subject.

A list-based publish/subscribe pattern requires to identify and to maintain a list of subscribers for a specific event; when the event occurs, each subscriber in the subscription list is advised. The idea, in Promise, is that users are automatically added to one or more lists by the system, according to their role(s). When a user creates or modifies a system object, all the users that are in the relevant lists will receive a notification about the event.

# 4 Promise functionalities supporting collaboration

We propose a collaborative interaction based on asynchronous methods. The usual tools used to support asynchronous collaboration are:

- E-mail
- Instant messaging
- Application sharing
- Videoconferencing
- Collaborative workspace and document management
- Task and workflow-management
- Wiki group or community effort to edit wiki pages (e.g., wiki pages describing concepts to enable a common understanding tithing a group or community)
- Annotations
- Feedback
- Messages
- Polls

While, in principle, all the above means can be used by the Promise users, we propose to integrate in the system a basic set of asynchronous interaction means, avoiding to duplicate already existing methods (e.g., e-mail). In particular we plan to rely on the following:

- Annotated resources
- Feedbacks
- Polls
- Messages

## 4.1 Annotated resources

All the relevant data managed by the Promise system is modeled through the notion of annotated resource (e.g., a topic, a document, a set of measures, etc.). Users, according to their roles and privileges, can create, modify, and annotate such resources that constitute the backbone of the collaboration system.

## 4.2 Polls

The users will able to set up one or more polls. With this tool the users can collect other collaborators' opinion about whatever they want.

## 4.3 Messages

The users can send simple private message. This is the simplest way to ask or talk about something with a specific user.

## 4.4 A simple example scenario

Bobby is in charge of selecting the topics for an evaluation a campaign. He creates a new topic called "Romans"; when he saves the topic he labels the topic with one or more subjects. The information about this event is sent to all the pertinent users (the users that are registered in the associated subjects, users that are organizing the campaign, users that are in charge of selecting topics for that campaign, etc.). One of them, John, reads the topic and comes up with the idea that the topic name is wrong: he puts an annotation on the topic and creates a poll to choose among three other names for the new topic. The poll invitation is sent to all relevant users. The poll ends, and according to the collected opinions the topic's name is changed in "Roman Empire". So John modifies the topic, renames it in "Roman Empire" and inserts an annotation about the poll and the old name of the topic. This is again sent to all the relevant users, which can insert new annotations or propose new changes.

# 5 Collaboration and knowledge sharing user requirements

The section structure follows the IEEE 830-1998 Recommended Practice Software Requirements Specifications requirement standard described in Appendix 8.3. To make the structure more clear, subsection numbering will be the same of the one presented in the Appendix 8.3. That allows for using this section out of the context of the whole report.

## 5.1 Introduction

### 5.1.1 Purpose

This section contains a summary of the collaboration and knowledge sharing Promise user requirements. The section structure follows the IEEE 830-1998 Recommended Practice Software Requirements Specifications requirement standard described in Appendix 8.3.

Collaborative software has the goal of helping people involved in a common task to achieve their goals. We distinguish two main levels:

1. Knowledge-sharing, and
2. Collaborative interaction.

With the expression knowledge-sharing we mean that users can share information about their work (insight, useful data, interesting visualization, etc.). With collaborative interaction we refer to the system supported functionalities that facilitate collaboration among users.

### 5.1.2 Scope

The software that we want to produce is a "Collaborative User Interface" that addresses two main issues: knowledge-sharing and collaboration. The objective is to provide users with an active collaboration tool, to enrich and augment the acquired knowledge-base with interpretations, and additional information. Referring to the issue of knowledge-sharing, this tool should provide users a way in which they can share their own achievements as well as being updated about others' progresses.

### 5.1.3 Definition, acronyms, and abbreviation

CUI - Collaborative User Interface

REST - Representational State Transfer

R-USER - User with a specific role

SRS - Software Requirements Specification

### 5.1.4 References

- Portlet guide on sun/oracle web site - http://download.oracle.com/docs/cd/E14571_01/webcenter.1111/e10148/jpsdg_intro_portlets.htm
- REST article on sun/oracle web site - http://www.oracle.com/technetwork/articles/javase/index-137171.html
- Handouts on Professor Baldoni's web site on publish/subscribe paradigm. - http://www.dis.uniroma1.it/~baldoni/SDslide_pubsub_2008_SD.pdf

### 5.1.5 Overview

The main purpose of this section is to describe how the CUI supports the cooperation within PROMISE community. In the next sections we will see an overall description of the CUI, focusing on user interface, interfaces with other applications, and a general characterization of its functionalities. We also provide interaction details, explaining how the users interact with the CUI.

## 5.2 Overall descrition

### 5.2.1 Productive perspective

The "Collaborative User Interface" is a component of the PROMISE environment. In particular it addresses the need of developers to make the PROMISE open evaluation infrastructure a kind of virtual research environment, where the whole process which leads to the creation, maintenance, dissemination, and sharing of the knowledge produced during an evaluation campaign is taken into consideration and fostered, and an active communication vehicle for the communities interested in the experimental evaluation. This will be achieved by offering advanced annotation and collaboration functionalities in order to become not only the place where storing and accessing the experimental results take place, but also an active communication tool for studying, discussing, comparing the evaluation results, where people can enrich the information managed through it with their own annotations, tags, and share them in a sort of social evaluation community.

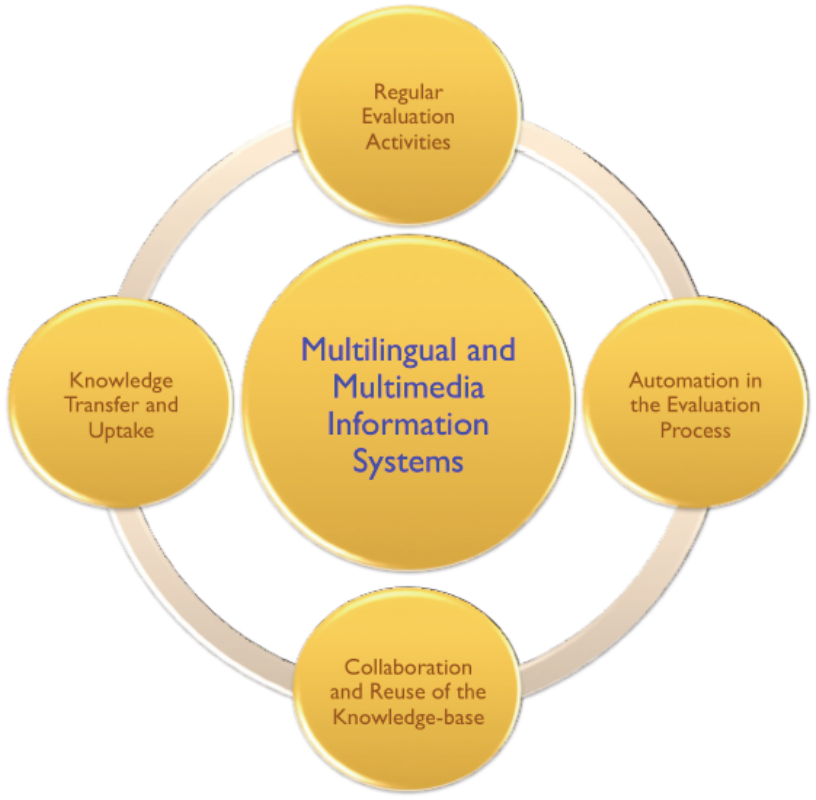

Regular Evaluation Activities
Knowledge Transfer and Uptake
Multilingual and Multimedia Information Systems
Automation in the Evaluation Process
Collaboration and Reuse of the Knowledge-base

**Figure 1: Approach adopted by PROMISE**

#### *5.2.1.1 System interfaces*

To build the CUI we foresee to use the portlets architecture. A portlet is a reusable Web component that can draw content from many different sources. Portlets provide a means of presenting data from multiple sources in a meaningful and related way. Portlets can display excerpts of other Web sites, generate summaries of key information, perform searches, and access assembled collections of information from a variety of data sources. Because different portlets can be placed on a common page, the user receives a single-source experience. In reality, the content may be derived from multiple sources. The final representation of page is entrusted to a portlet container which task is to follows the user action, execute the business logic of all portlets, and aggregate the result that each portlet produces in a single page.

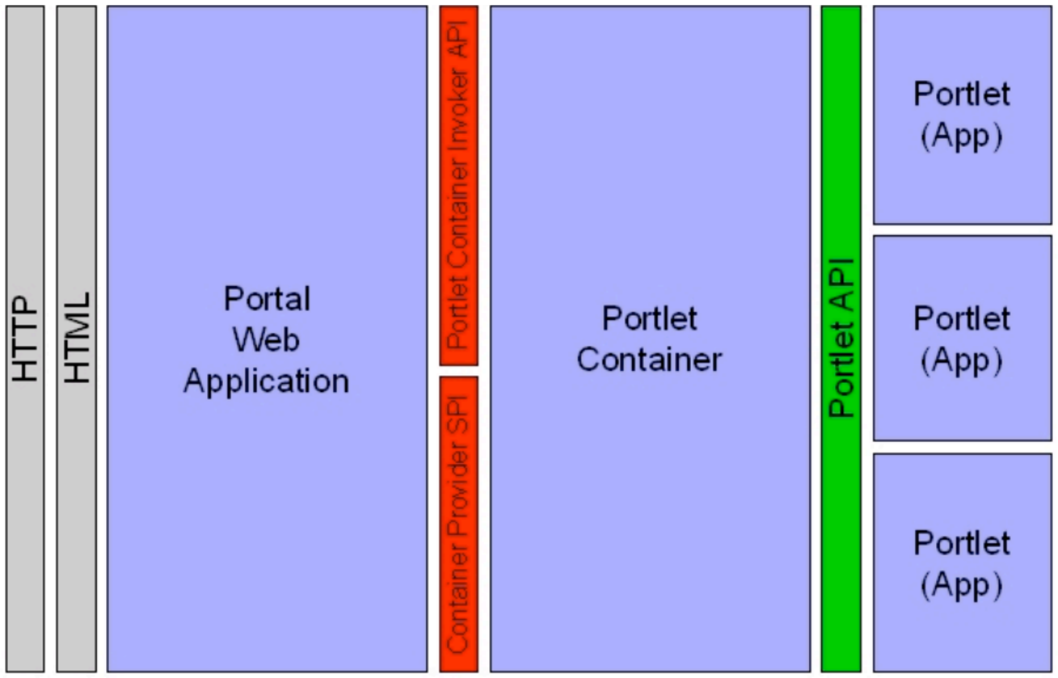


**Figure 2: Example of portlets architecture**

#### *5.2.1.2 User interfaces*

The CUI consists of several modules: that allows a user to manage her/his personal data and to access the system resources. A suitable menu contains all the foreseen tasks and a data section displays the relevant data for the task at hand. In particular, the menu provides the following choices:

1. "Personal information": when a user selects this option the system shows his personal information, like complete name, email, username, password, role(s) in the system, and nationality. The user can change his personal information.
2. "News": this option allows for accessing the news concerning the user's work, like notification about documents of his own interest, new messages, poll requests, creation of new groups or subjects. The user has the possibility to delete news.
3. "Subscriptions": when a user selects this option, the data section shows the groups or the subjects which he joined, and his favorite documents. The user can choose to unsubscribe from a group, or a subject, and to delete a document from his favorite list.
4. "Community": when a user selects this option he can see, in the data section, other groups and topics of the community. In this section the user can join a group or a subject.
5. "Messages": when a user selects this option he can see, in the data section, his private conversation with other users of the community. Conversations are grouped by contacts. The data section provides a mechanism to highlights the unread

message. The number of unread messages is also reported next to the message command of the option menu. The user can sends messages to other users or other groups of users, read messages that he has received, and reply to them.

6. “Wiki”: when a user selects this option he can see, in the data section, wiki pages used by community to discuss and to reach agreement on various issues. The user can see an existing wiki page and, in case, comment it; s/he can add a new wiki page or modify a previously created wiki page.
7. “Polls”: when a user selects this option he can see, in the data section, the polls that have been proposed in the community. A poll can be visible to all users or to restricted group of users. The user can reply to a poll or can start a new poll.

#### *5.2.1.3 Software interfaces*
Not yet available.

#### *5.2.1.4 Communication interfaces*
In the above section we mentioned that the CUI is based on a portlets architecture and how it is part of the Promise project. In this section we will describe how the CUI communicates with the rest of the system. In the underlying level there is a service logic which uses a restful web service. A restful web service is a simple web service implemented using HTTP and the principle of REST. In the web services world, Representational State Transfer (REST) is a key design idiom that embraces a stateless client-server architecture in which the web services are viewed as resources and can be identified by their URLs. Web service clients that want to use these resources access a particular representation by transferring application content using a small globally defined set of remote methods that describe the action to be performed on the resource. REST is an analytical description of the existing web architecture, and thus the interplay between the style and the underlying HTTP protocol appears seamless.

#### *5.2.1.5 Operations*
The CUI allows users to perform their tasks according with their roles and granted permissions. The following is a list of tasks that all the users should be able to perform:

1. To login
2. To modify the password.
3. To view news.
4. To view the list of subscriptions.
5. To add documents in some specific areas.
6. To add a subject in some specific areas.
7. To join other groups of interest. Note that this task allows users to receive information about document concerning these groups, not to belong to them.

8. To unsubscribe from a group. Notes that the system prevents user from unsubscribe from the groups relevant to their roles.
9. To join a particular document of interest.
10. To unsubscribe from a document.
11. To join particular subject of interest.
12. To unsubscribe from a subject.
13. To send private message.
14. To send message to a group.
15. To read message.
16. To delete message.
17. To delete conversation.
18. To add a wiki.
19. To comment a wiki.
20. To modify a wiki that he previously inserted.
21. To start a poll restricted to a group.
22. To start a public poll.
23. To participate in a poll started by other users
24. To annotate a resource.

In addition to previous list, there are some tasks that only users with special permission, can perform. These tasks are the following:

1. To assign a role to a user
2. To change a role of a user.
3. To create a new role.
4. To delete a role.
5. To create a subject.
6. To delete a subject.
7. To delete a document.

#### 5.2.1.6 Site Adaptation requirements

All modules must be portlets, so they are inherently portable: they need only a portlet container to run.

### 5.2.2 Product Functions

The CUI should provide users with the following functionalities:

1. A mechanism to authenticate users and to assign them role(s) within the community. The roles must be chosen from the following (see Section 2): organizer, participant, relevance assessor, topic creator, site administrator, other researcher, and annotator. In the following, if we will not have to specify differently, we will refer to members of community with name R-User, which indicate a member of community who has one or more of above roles.
2. A mechanism to create groups based on roles defined within the community.

3. A mechanism to add automatically the users to a group in a manner consistent with their roles.
4. A mechanism to inform the users who joined a particular group about the addition, modification, or annotation of documents relevant for that group. This mechanism follows the list-based publish/subscribe paradigm in which there is a list of subscribers for a specific event. When the event occurs the system informs all the subscribers in subscription-list. In this case the events are additions, modifications or deletions of documents and the lists of subscribers are groups. A publisher is a R-user, who adds, modify or delete a document which concerns a group.
5. A mechanism to inform the users who joined a particular subject about the additions, modifications or deletions of documents involving that subject. This mechanism follows the subject-based publish/subscribe paradigm in which events (e.g., annotations or document creations) are "tagged" with one or more identifiers (subjects) when are published. Subscribers issue subscriptions according to the subjects they are interested in. A subject can be thus represented as a "virtual channel" connecting producers to consumers. In this case the subjects are the arguments to which a document pertains and the subscribers are R-Users who joined those subjects. A publisher is a R-user, who adds, modify, or delete a document which pertain to a particular subject. When one of these events occurs the system notifies all the subscribers.
6. A mechanism to inform the users who join a particular subject about the history of that subject, i.e., the relevant past events.
7. A mechanism to inform the administrator that a document has been translated into another language.

The collaboration includes two aspects:

- how users can work on resource together;
- How users can share each other opinions, doubts, ideas and insights, or simply coordinate their work.

About the first aspect the idea is to manage shared resources that users can modify asynchronously. The CUI provides users with the following functionalities:

1. An annotation mechanism that allows users to add, deletes, and modifies a note. Note that annotating a document is equivalent to changing it, so this mechanism triggers the dissemination mechanism described above.
2. A revision mechanism that maintains the consistency among the versions of a resource.
3. An access control mechanism that prevents from changing a resource simultaneously.

About the second aspect the CUI provides the users with a communication mechanism which includes the already discussed means: messages, wiki pages, and polls.

### 5.2.3 User characteristics

PROMISE will involve world-wide large researcher and developer communities with multidisciplinary competencies in its regular evaluation activities. The "Collaborative user interface" has to support the cooperation between these communities, so users who use this software are expert and motivated.

### 5.2.4 Constraints

The interfaces must be developed as web applications, so the developers should keep in mind the limit of web applications respect standalone applications, and compatibility with the available browsers (Internet Explorer, Firefox, Safari, Opera ...).

### 5.2.5 Assumptions and dependencies

The CUI is entirely developed using the Java technology and portlets: the unique dependency is the availability of a portlet container.

## 5.3 Specific requirements

### 5.3.1 External interfaces

Not yet defined.

### 5.3.2 Functional requirements

In this section we will show some practical scenarios to illustrate how the users can use the CUI.

**Scenario 1: Creating an R-user**

Suppose that Bob wants to create an account. He has to request his credential, compile the form with his personal information and submit them to the system. Suppose that Bill is an R-user, who has the permission to assign a role to other members of community. When Bill receives the Bob's request, he has to verify his data. There are two possibilities:

1. the data are correct;
2. The data are incorrect.

In the first case Bill assigns an existing role(s) to Bob and creates the account. Suppose that Bill assign to Bob the roles of annotator and topic creator. The system sends a confirmation email to Bob with his username, password and roles. The system also adds Bob to the annotators group and to the topic creator's group.

In the second case Bill does not create the account and Bob receives an email with the motivation of this decision.

**Scenario 2: an event which interests particular group occurs**

Suppose that Bob is a topic creator and that he modifies the document "Topics for Evaluation Campaign X, 2011" which is relevant to the topic creator group. When the system notices that the modification event occurs, verifies which groups are interested in this event and sends a message to all users who joined these groups.

**Scenario 3: an event that pertains to a particular subject occurs**
Suppose that Alice and Bob belong to the organizers group and that they join the subject "CLEF 2011". Suppose also that Bob add the document "Evaluation Campaign Guidelines" which is tagged with subject "CLEF 2011". When the system notices that the addition event occurs searches all users who joined subject "CLEF 2011" and sends a message to inform them that Bob added the document "Evaluation Campaign Guidelines".

**Scenario 4: manual subscription to a group**
Suppose that Bob is registered to the system with the role of topic creator and that he is interested in the work of other researchers. To join to this group he has to go in the community section, select the group "other researcher" and join to it. The system sends to Bob a confirmation message that appear in the news section of "User Module" and a confirmation email to his email address. If Bob go into subscription section of "User Module" he can note that under groups of interest appears, as well as "topic creator", "other researcher". It is worth noting that Bob is automatically added from the system to topic creators group.

**Scenario 5: automatic subscription to a subject**
Suppose that Bob is modifying the document "Evaluation Campaign Guidelines" and that this documents pertains to the subject "CLE2011" which is not among those to whom Bob is subscribed. When he saves his changes the system ask to him if he want to subscribe to the subject "CLE2011". He accepts, and the system asks Bob if he wants to see the history of the subject.

**Scenario 6: manual subscription to a subject**
Suppose that an organizer creates a new subject called "CLE2011". Suppose also that Bob is interested in the new subject. To join this subject he has to go in the community section, select it and join to it. The system sends to Bob a confirmation message that appear in the news section of "User Module" and a confirmation email to his email address. Moreover, the system asks to Bob if he wants to see the history of the subject. If Bob go into subscription section of "User Module" he can note that the list of subject of interest now includes "CLE2011".

**Scenario 7: un-subscription from a group**
Suppose that Bob is a topic creator and that he is subscribed to the other researchers group. To unsubscribe from this group he has to go in the subscription section of the "User Module", select "other researcher" and delete it. Note that the system prevents user to

delete the groups automatically assigned by role. So if Bob try to delete “topic creator” the system inform him that this operation is not possible.

**Scenario 8: un-subscription from a subject**
Suppose that Bob subscribed the “CLE2011” subject. To unsubscribe from this subject he has to go in the subscription section of the “User Module”, select “CLE2011” and delete it.

**Scenario 9: conversation**
Suppose that Bob wants to ask for an opinion to Bill about the next evaluation campaign. He can choose “Message” in the option menu and send a message to Bill. If Bill goes in his personal page after receiving a message he can see that next to the message voice of the option menu there is at least one unread message. Suppose that Bill reads the Bob’s message and then reply to Bob using the reply option. When Bob receives the response he can see all conversation between him and Bill.

**Scenario 10: wiki**
Suppose that the organizers produce the document “Guideline for new Evaluation Campaigns” and those they want to seek advice on the content of document to the other members of community. Suppose also that Bob is the organizer who is responsible for collecting opinions within community. He can choose “Wiki” in the option menu and add a new wiki page with content of document. Now suppose that Alice is a topic creator and that he has perplexity on “Guidelines for new Evaluation Campaigns”. She can show her doubts adding a comment to wiki.

**Scenario 11: poll**
Suppose that Bob is an organizer and that he wants to ask for an opinion to other members of his group. He can choose “Polls” in the option menu and select “promote restricted poll”. Then he has to insert the group(s) which can see the poll (e.g., “Organizers Group”), the object of poll (e.g., “Evaluation Campaign”) and the first message in the poll. The system sends to all user involved by Bob both a message and an email to inform them. Suppose that Bill reply to the poll “Evaluation Campaign”. Bob and the other user to which the poll is visible receives message (and also an email) which inform them that Bill replied to the poll “Evaluation Campaign”. If Bob wants to see the response he can choose “Polls” in the option menu and select “Evaluation Campaign”.

**Scenario 12: annotation**

Suppose that Bob is an organizer and that he wants to modify the document "Guidelines for new Evaluation Campaigns". He has to make his changes, in case adding one or more notes. For each note he must specify the visibility of note (public, private, particular group). Now suppose that another organizer, Alice, attempts to modify the document. The system prevents Alice to do this operation and inform her that another user is modifying the document. When Bob end his work he saves the changes. The system saves both the copy without the Bob's changes and the copy with Bob's changes. The system also saves the annotation that were added from Bob adding to them date of modify and user who did the modification. Then the system sends a message to all users who are interested in the document to inform them that a modification occurred.

### 5.3.3 Performance requirements

The system should support a high number of simultaneous users, and must be able to store a huge amount of heterogeneous data, constituted by all the annotations saved by the users (text, graphs, queries etc.).

# 6 Collaboration and knowledge sharing in visual analytics

The section structure follows the IEEE 830-1998 Recommended Practice Software Requirements Specifications requirement standard described in Appendix 8.3. To make the structure more clear, subsection numbering will be the same of the one presented in the Appendix 5.3. That allows for using this section out of the context of the whole report.

## 6.1 Introduction

### 6.1.1 Purpose

This section contains a summary of the collaboration and knowledge sharing Promise user requirements specific to the Visual Analytics component. The section structure follows the IEEE 830-1998 Recommended Practice Software Requirements Specifications requirement standard described in Appendix 8.3.

In order to better clarify the collaborative functionalities that Promise will provide in such a context this section presents some preliminary requirements about the Visual Analytics user interface. The full set of requirements will be presented on Deliverable 5.2 - User interface and Visual analytics environment requirements (month 12).

### 6.1.2 Scope

This software component is the part of the Collaborative User Interface (CUI) aiming at to realizing collaboration and knowledge sharing for users that use the Visual Analytics environment.

Annotations are crucial to the extent of reconstructing the operations leading to a visualization. Through annotations one can explain executed operations and can explain, spread, and save particular choices. The same holds for queries executed by the system during the mapping process (the process leading from data to the visualization).

Moreover, annotations are useful for Promise as a means of remote and asynchronous communication among users: e.g., one can use them to express his opinion about the visualizations produced by other users.

Annotations are also useful to improve the quality of a work, indeed one can use them to explain the modifications executed on works previously produced (by himself or by other users), and obtain positive or negative feedbacks and suggestions by others.

The annotation mechanism is integrated with knowledge sharing: indeed every user interested in a subject/event (a subscriber) receives a notification every time a related object is annotated (published).

### 6.1.3 Definitions, acronyms, and abbreviations

Mock-up definition: it is a model of a design used for demonstration, design evaluation and other purpose. A mock-up is called a prototype if it provides at least part of the functionality of a system and enables testing of a design.

CUI - Collaborative User Interface

REST - Representational State Transfer

VA - Visual Analytics

### 6.1.4 References

Not presents.

### 6.1.5 Overview

In the following there will be:

- An overall description of the system functionalities, considering annotation mechanism into a wider context (visualization). Here no specific requirement is formulated; rather there is a brief overview of some factors affecting it.
- A more detailed description of the required functions.

## 6.2 2 - Overall description

### 6.2.1 Productive perspective

The software is a component of the Promise system. The visual analytics tools give a useful support for analyzing the results of an experiment and collaboration is a key issue in such a context. Details about the software architecture will be presented on Deliverable 5.2 - User interface and Visual analytics environment requirements (month 12).

#### 6.2.1.1 System interfaces

To use visual analytics tools, we rely on the portlets architecture. A portlet is a reusable web component that can draw content from many different sources. Different portlets can be displayed on a web page. So we can have multiple visualizations on a single web page.

#### 6.2.1.2 User interfaces

The visual analytics tool consists of different modules. These modules allow the user to manage and view some data with different visualizations.

The designer should take into account the annotation mechanism nature: it has to make easy the work of skilled users belonging to a collaborative group and annotations should be considered from a visualization point of view.

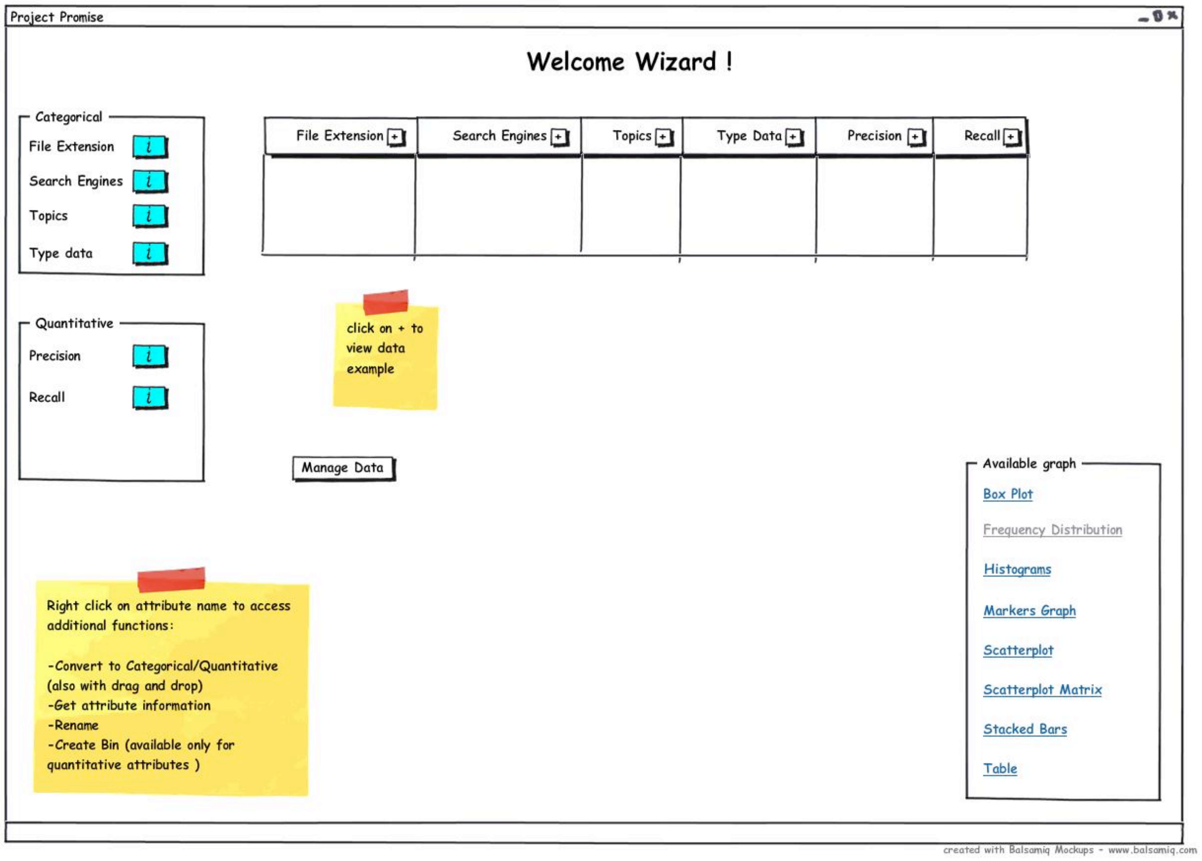
In order to show the usefulness of this mechanism, a brief overview of the (to be) visualization system is provided. It doesn't belong to this deliverable, but its general structure is needed to describe and understand how annotations work in such a context. As a support to this discussion, also some preliminary mock-ups are provided.

In order to obtain a significant visualization of a huge amount of heterogeneous data according to the user's goals, the system will provide four main interfaces: the home page of the wizard, the module devoted to data manipulation (data managing), the module realizing the mapping (from data to visualization), and the module for the ultimate data filtering.

In general, data are tuples coming from (relational) tables and the user is allowed for exploring data features, by means of a sampling operation. Tuples are presented to the user in form of a table, and, when needed, the table structure can be rearranged and modified through suitable operations (mathematic or of grouping/reordering) producing a different table, derived from the "original" one.

The above interfaces take different kind of input and provide different kind of output. In particular:

- The home page takes a list of attributes as input, and provides the same list as output, but with a (possibly) different classification (into quantitative and categorical) of the attributes.



click on + to view data example

Right click on attribute name to access additional functions:

-Convert to Categorical/Quantitative (also with drag and drop)
-Get attribute information
-Rename
-Create Bin (available only for quantitative attributes )

**Figure 1: the home page mock-up**

- The interface, devoted to data manipulation, takes as input a table (organized in an arbitrary way) and provides a different table as output, where data are rearranged, and with possibly more columns deriving from the “original” table, through some mathematical or ordering/grouping operation. The manipulated table must be suitable for the user visualization purposes (otherwise, some warning messages will come up, leading the user to a more compliant data organization).

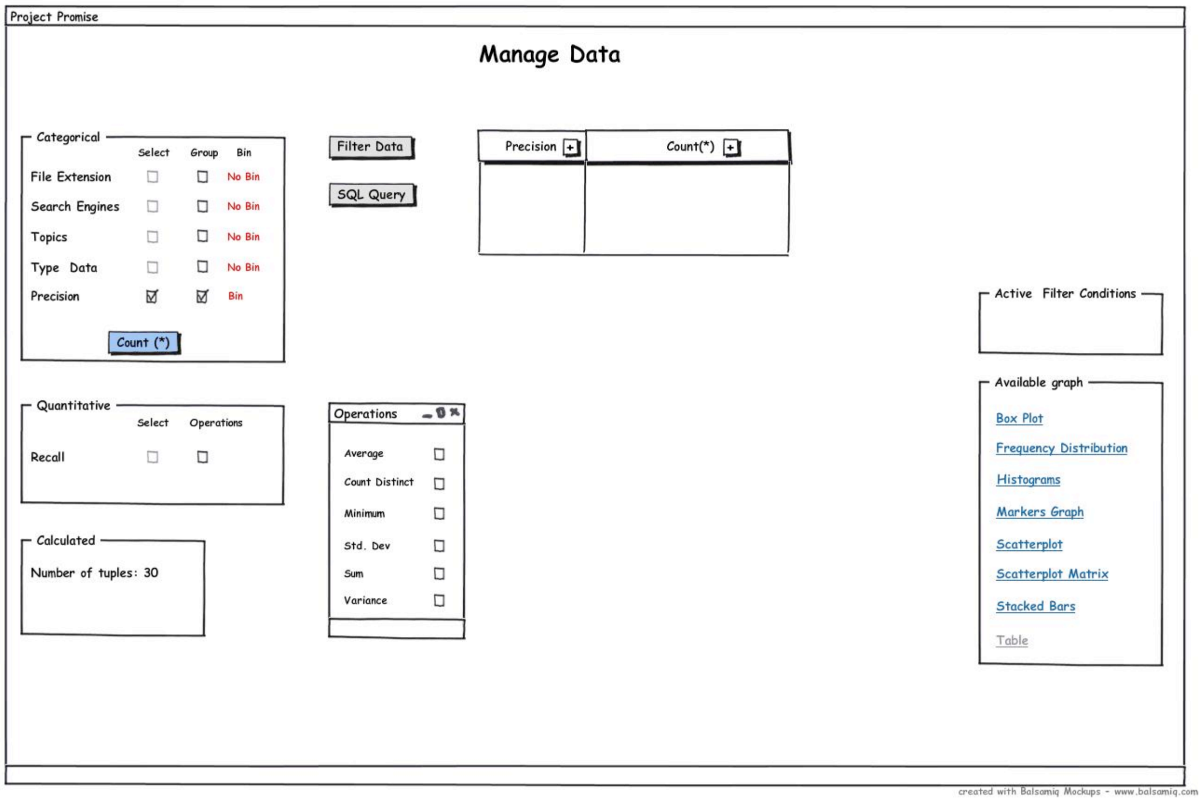


**Figure 2: the data manipulation interface mock-up**

- The interface, realizing the mapping, takes as input the table (manipulated, when needed) and produces as output a graph representing the visualization; in the following, as an example, there is a preliminary mock-up about a scatterplot, but the same holds for all kinds of visualization.

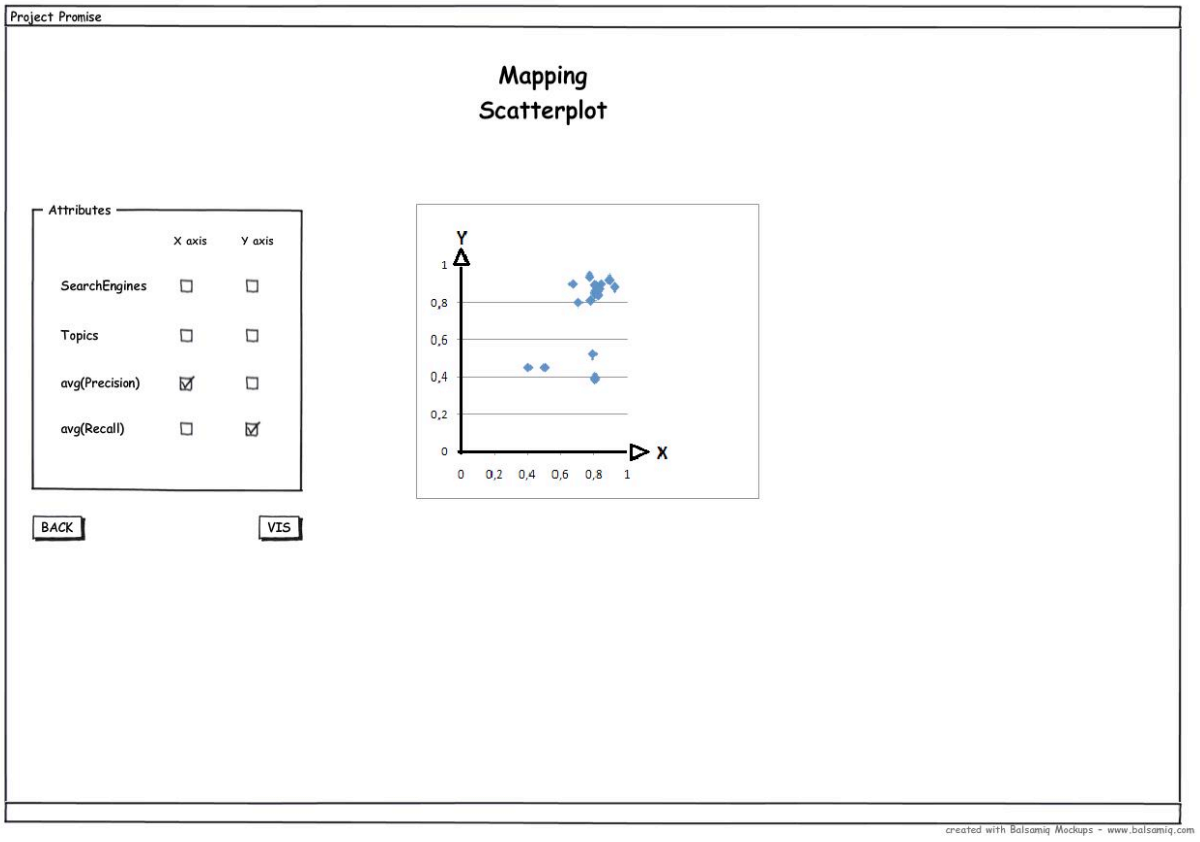


**Figure 3: the mapping interface mock-up**

- The filter interface takes as input a set of data coming from a table (manipulated or original), and provide as output a subset of the same data.

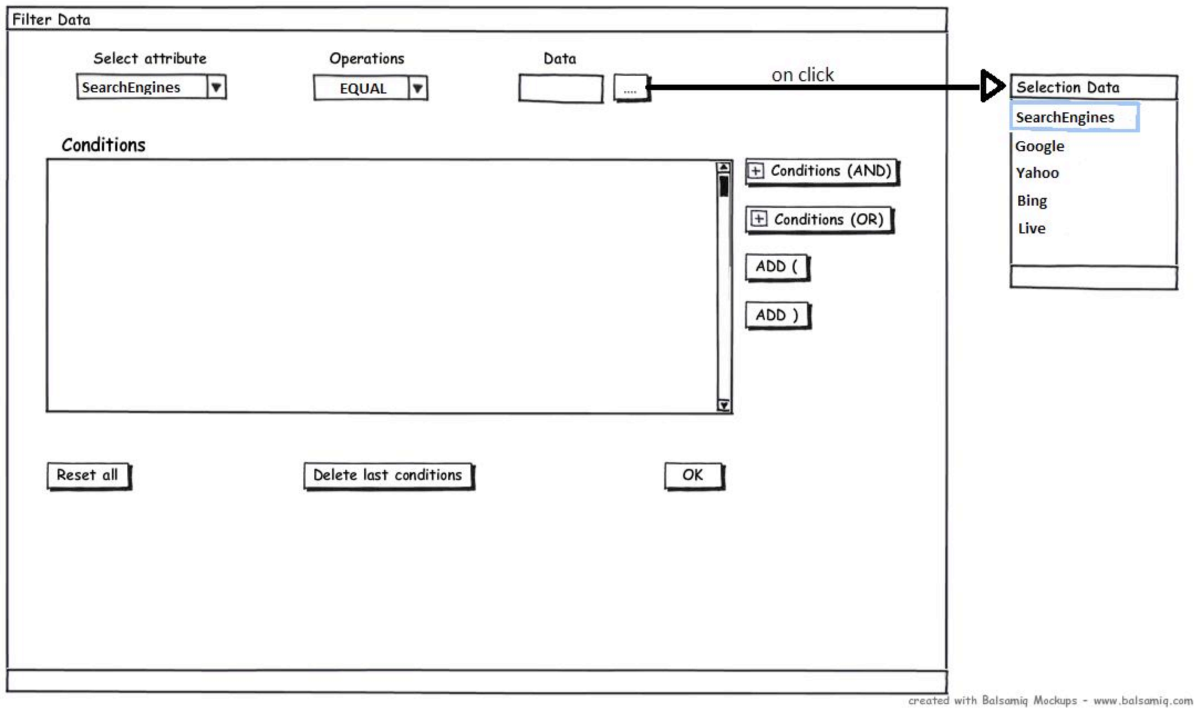


**Figure 4: the filter interface mock-up**

Obviously, the above interfaces are part of a sequence of operations, so there is a sort of chain where the output produced by each of them constitutes the input of the following one.

The user expresses the commands by dragging attributes from categorical to quantitative and vice versa, selecting operations by means of checkboxes and slide bars. The greater part of them will be implemented through suitable queries.

The system shall have as "first" input a correct "separation" between categorical and quantitative attributes, because it needs to have a clear distinction for the operations in the rest of the process.

Then, according to this distinction, the user can choose the operations (group by, sum, ordering, filtering, etc.) to organize the data, and he/she can manipulate the original table, in order to obtain a suitable table for his/her purposes. Obviously, if data organization satisfies the user, he/she can skip this.

Then, the user can choose the visualization, with some other adjustment in the data set.

If a particular data set leads to a non-significant visualization, same suitable warning messages will notify this no good situation to the user, with also some indications about what is wrong with it.

#### *6.2.1.3 Software interfaces*
Not yet applicable.

#### *6.2.1.4 Communications interfaces*
The collaboration in visual analytics is based on portlets architecture. The portlets communicate each other and with rest of the system with REST web services.

#### *6.2.1.5 Operations*
When a user annotates some object, the system should save it with the associated comment. After this operation, the knowledge sharing system should notify the annotation to all the users (subscribers) interested in that argument.

#### *6.2.1.6 Site adaptation requirements*
All modules must be portlets, so they are inherently portable: to be executed they only need a portlet container.

### 6.2.2 Product functions
To solve the problem of the knowledge-sharing and collaboration, the visual analytics tools provide users with the following functionalities:

1. A mechanism to manipulate the data associated with the Promise system.
2. A mechanism to create visualizations (scatterplots, graphs, etc.).
3. A mechanism to add annotation about visualizations and data manipulation.
4. A mechanism to inform the interested users that one or more visualizations have been created.

### 6.2.3 User characteristics
The final user is a technical expert, skilled, and with a high level of experience.

### 6.2.4 Constraints
The interfaces must be developed as a web application, so the developers should keep in mind the limit of web applications respect standalone applications, and compatibility with available browsers (Internet Explorer, Firefox, Safari, Opera, etc. ...).

### 6.2.5 Assumptions and dependencies

The CUI is entirely developed using the Java technology and portlets: the unique dependency is the availability of a portlet container.

## 6.3 Specific requirements

### 6.3.1 External interfaces

As stated above, there will be no particular annotation interface, but there will be a specific command to select in order to “register” the annotation, with a corresponding text space to add a comment (a simple consideration or an important observation) to the annotated object.

### 6.3.2 Functional requirements

The system should register the annotation with the associated text constituting the comment, and at the same time the annotation should be notified to all the users interested in that given argument. This means that the annotation mechanism should be integrated in the knowledge sharing system.
Given the annotation mechanism nature, in order to show its functions the following simple storyboard is presented:

Alice is a skilled user, looking for a correlation between the employees’ country and salary.

She accesses a table whose attributes contain the name of the employee, the salary, and the original country. Being a skilled user she knows that a scatterplot is a good way of investigate correlations, so she’s aware that data are not suitable for her purpose: she has to manipulate it.

She proceeds to a group by (based on country) operation, but she wants to annotate this operation because she wants notify to every other user her choice, in order to help them to reconstruct her visualization process. For the same reason, when Alice decides to activates a K-means clustering algorithm, she annotates also this operation (in this way annotations in visual analytics are accomplished).

Before visualizing the data, she decides to investigate only high salary, so she filters the manipulated data, in order to work only on salaries greater than a threshold. Obviously, she annotates also this decision for the same reason stated above.

At the end she visualizes the data, and she wants to add her consideration to the visualization, so she annotates her impressions, in order to communicate them to the other users working at the same project.

Later on, Bob wants to examine the Alice’s work, so he retrieves the visualization she has produced. In doing so, he is able to reconstruct exactly her mental process. Indeed, he can analyze her work, without any waste of time, only looking at her annotations. He has no doubt about what data she has visualized, and about what manipulation she has done on such data.

He is also able to read her final considerations, and to communicate his own opinions about her work and her conclusions through another annotation (this communication by means of annotations, in particular, accomplishes collaboration).

All the annotations are notified also to all the other users working at the same project and with a suitable role, such as other participants, interested in that particular argument (this accomplishes integration of collaboration and knowledge sharing).

### 6.3.3 Performance requirements

The system should support a high number of simultaneous users, and must be able to save and retrieve a huge amount of heterogeneous data, constituted by all the annotations saved by the users (text, graphs, queries etc.).

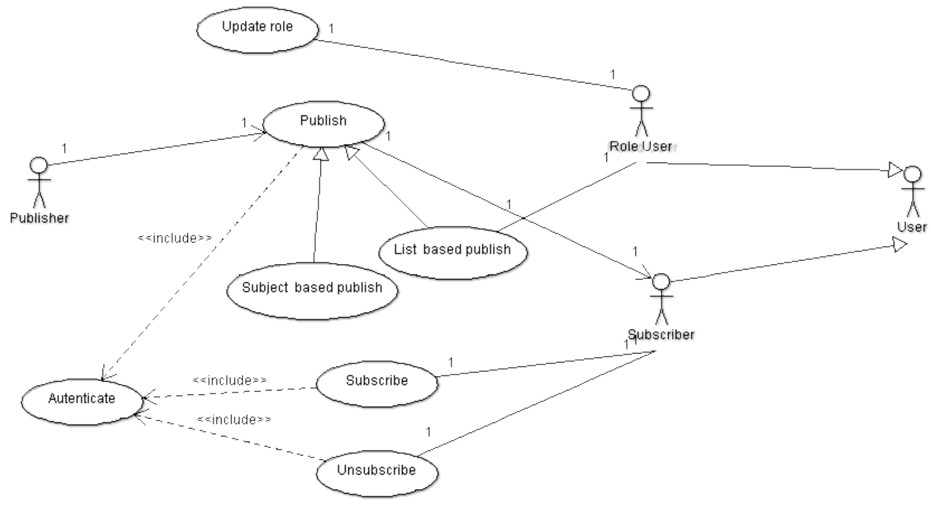
# 7 UML Use Cases

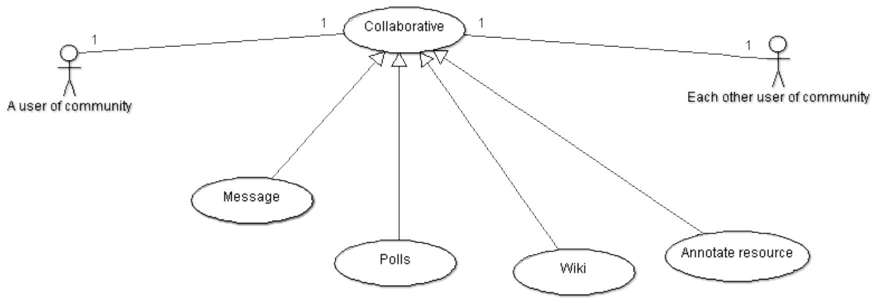
The actors described in WP3, see Appendix 5.1, play quite different roles with respect to the Promise environment; however, the knowledge sharing and collaboration functionalities induce on them an orthogonal classification. As an example, consider the activity of distributing the information that a new document has been released to a list of user according to their interest in a subject. That can be used for sharing a new topic creation, or a new visualization, or the result of a new experiment run. The same holds for annotations: the goal of an annotation could be quite different, but the system functionalities will be the same.

According to the requirements described in the previous sections, the following actors have been identified for the Promise UML Use Cases describing collaboration and knowledge-sharing.

- **User**. This actor is a generic Promise user
- **Role User**. This actor is a Promise user that is interacting with the system with one or more specific roles (according to the list presented on Appendix 5.1).
- **Subscriber**. This actor is a Promise user that requests the system to notify him of all the events that are pertinent to one or more subjects (e.g., a CLEF track).
- **Publisher**. A publisher is a data producer/modifier (either a human being or a software system). Each time an object is created, annotated, or modified, this information is distributed to pertinent role users or subscribers.
- **Community member**. A community member is a Promise user that explicitly works within a community with common goals (e.g., a group of experts that are creating topics for a specific evaluation campaign).

The following diagrams, produced with the ArgoUML (ver. 0.30.2) tool that supports UML notation version 1.4., show the UML Use Cases that involve such actors.





| | | | |
|---|---|---|---|
| **Use Case UML ID:** | PR1 | | |
| **Use Case UML Name:** | Publish | | |
| **Primary Actor:** | Publisher | | |
| **Secondary Actor(s):** | Subscriber | | |
| **Description:** | This UML Use Case allows users for sharing information about their work (insights, useful data, interesting visualizations, etc.) | | |
| **Trigger:** | Publishers produce their information | | |
| **Preconditions:** | At least a user is subscribed | | |
| **Postconditions:** | Each subscribed user read published information | | |
| **Normal Flow:** | | Actor Input | System Response |
| | 1 | | The system verifies the publisher authentication |
| | 2 | Publisher publishes a new event/document | |
| | 3 | | System notifies subscribers abou new event/document |
| **Exceptions:** | - | | |
| **Include:** | Authenticate | | |

**Table 1**

| | |
|---|---|
| **Use Case UML ID:** | PR2 |
| **Use Case UML Name:** | Update role |
| **Primary Actor:** | Administrator |
| **Secondary Actor(s):** | Role User |
| **Description:** | System assigns role(s) to user |
| **Trigger:** | Administrator assigns role(s) to user. A user is subscribed |
| **Preconditions:** | An existing user |

| | |
|---|---|
| | An existing role |
| **Postconditions:** | The user is assigned with the new role(s) |
| **Normal Flow:** | |

| | Actor Input | System Response |
|---|---|---|
| 1 | The system administrator selects a user | |
| 2 | The System Administrator assigns one or more roles to the user of point 1 | |
| 3 | | The system assigns role(s) to the user |

| | |
|---|---|
| **Exceptions:** | - |
| **Include:** | - |

**Table 2**

| | |
|---|---|
| **Use Case UML ID:** | PR3 |
| **Use Case UML Name:** | Authenticate |
| **Primary Actor:** | -A Role User |
| **Secondary Actor(s):** | - |
| **Description:** | The system verifies the credentials of a user |
| **Trigger:** | When a service (a UML Use Case, that includes it) starts |
| **Preconditions:** | - |
| **Postconditions:** | The user's credentials are verified by the system |
| **Normal Flow:** | |

| | Actor Input | System Response |
|---|---|---|
| 1 | | System reads username and password (typed in by the user) |

| | |
|---|---|
| | 2 \| \| System verifies the credentials of point 1 |
| **Exceptions:** | - User or password are null<br>- User or password are incorrect |
| **Include:** | - |

**Table 3**

| | |
|---|---|
| **Use Case UML ID:** | PR4 |
| **Use Case UML Name:** | Subject-based publish |
| **Primary Actor:** | Publisher |
| **Secondary Actor(s):** | Subscriber |
| **Description:** | The *subject-based publish* is a particular kind of publish Use Case, in which events (e.g., annotations) are "tagged" with one or more identifiers (subject) during their publication. |
| **Trigger:** | - |
| **Preconditions:** | A user is subscribed<br>A user has tagged his subjects |
| **Postconditions:** | - |
| **Normal Flow:** | Point 3 of UML Use Case, labeled Publish.<br><br>\| \| Actor Input \| System Response \|<br>\| 1 \| \| The system finds events for the publisher's subjects \|<br>\| 2 \| \| The system notifies subscribers of a new event/document \| |
| **Exceptions:** | - |
| **Include:** | - |

**Table 4**

| | |
|---|---|
| **Use Case UML ID:** | PR5 |

| | |
|---|---|
| **Use Case UML Name:** | List-based publish |
| **Primary Actor:** | Publisher |
| **Secondary Actor(s):** | Role User |
| **Description:** | The Use Case identifies and maintains a list of subscribers for a specific event; when the event occurs, each subscriber in the subscription list is advised. |
| **Trigger:** | Publishers publish their information |
| **Preconditions:** | A list of subscriber |
| **Postconditions:** | Each subscribed user, included in a list, reads published information |
| **Normal Flow:** | Point 3 of Use Case UML, labeled Publish. |

| | Actor Input | System Response |
|---|---|---|
| 1 | | System finds events/documents every subscriber belonging to a li |
| 2 | | System notifies to subscribers a r event/document. |

| | |
|---|---|
| **Exceptions:** | - |
| **Include:** | - |

**Table 5**

| | |
|---|---|
| **Use Case UML ID:** | PR6 |
| **Use Case UML Name:** | Subscribe |
| **Primary Actor:** | Subscriber |
| **Secondary Actor(s):** | - |
| **Description:** | It assigns to a user a group or a subject |
| **Trigger:** | A user request |
| **Preconditions:** | |

| | |
|---|---|
| **Postconditions:** | The assignment of an event or a document to a user |
| **Normal Flow:** | The system verifies authenticated user (include ‘Authenticate’ UML Use Case), belonging to a list. |

| | Actor Input | System Response |
|---|---|---|
| 1 | A user selects a group or a subject | |
| 2 | | The system assigns to the user a group or a subject to |

| | |
|---|---|
| **Exceptions:** | - |
| **Include:** | - |

**Table 6**

| | |
|---|---|
| **Use Case UML ID:** | PR7 |
| **Use Case UML Name:** | Unsubscribe |
| **Primary Actor:** | Subscriber |
| **Secondary Actor(s):** | - |
| **Description:** | It unsubscribes a user from a group or a subject |
| **Trigger:** | A user’s request |
| **Preconditions:** | - |
| **Postconditions:** | The de-assignment of a group or a subject |
| **Normal Flow:** | |

| | Actor Input | System Response |
|---|---|---|
| 1 | | System verifies an authenticated user (include ‘Authenticate’ UML Use Case), belonging to a list. |
| 2 | The User selects a group, subject | |
| 3 | | The System de-assigns a group |

| | | | |
|---|---|---|---|
| | | | or a subject |
| **Exceptions:** | - | | |
| **Include:** | - | | |

Table 7

| | |
|---|---|
| **Use Case UML ID:** | PR7 |
| **Use Case UML Name:** | Collaborative |
| **Primary Actor:** | A user of community |
| **Secondary Actor(s):** | Each other user of community |
| **Description:** | This Use Case describes the way of collaborate of users of community with software help |
| **Trigger:** | A user of community, who wants to communicate information |
| **Preconditions:** | A community with at least two users |
| **Postconditions:** | To create/communicate/share information (e.g., a document) |
| **Normal Flow:** | 1. Collaborative task (abstract) |
| **Exceptions:** | - |
| **Include:** | - |

Table 8

| | |
|---|---|
| **Use Case UML ID:** | PR8 |
| **Use Case UL Name:** | Message |
| **Primary Actor:** | - |
| **Secondary Actor(s):** | - |
| **Description:** | The users can send private message. |
| **Trigger:** | A user of community, who wants to communicate information |
| **Preconditions:** | - |
| **Postconditions:** | - |
| **Normal Flow:** | Point 1 of Use Case UML, labeled Collaborative |

| | Actor Input | System Response |
|---|---|---|
| 1 | A user send a message | |
| 2 | | System manipulates messages of point 1 |

| | |
|---|---|
| **Exceptions:** | - |
| **Include:** | - |

**Table 9**

| | |
|---|---|
| **Use Case UML ID:** | PR9 |
| **Use Case UML Name:** | Wiki |
| **Primary Actor:** | - A Role User |
| **Secondary Actor(s):** | - |
| **Description:** | Allows for creating/modifying Wiki pages. |
| **Trigger:** | A users of community, who wants to communicate information |

| | |
|---|---|
| **Preconditions:** | Wiki page isn't already in check out |
| **Postconditions:** | - |
| **Normal Flow:** | Point 1 of UML Use Case, labeled Collaborative |

| | Actor Input | System Response |
|---|---|---|
| 1 | The user create/modify a Wiki paget | |
| 2 | The user creates/modifies a Wiki page | |
| 3 | | The system stores the Page |
| | | |

| | |
|---|---|
| **Exceptions:** | - |
| **Include:** | - |

**Table 10**

| | |
|---|---|
| **Use Case UML ID:** | PR10 |
| **Use Case UML Name:** | Polls |
| **Primary Actor:** | - |
| **Secondary Actor(s):** | - |
| **Description:** | This UML Use Case allows a user for setting up one poll. |
| **Trigger:** | - |
| **Preconditions:** | - |
| **Postconditions:** | - |
| **Normal Flow:** | Point 1 of Use Case UML, labeled Collaborative |

| | Actor Input | System Response |
|---|---|---|
| 1 | A user creates one poll | |

| | | | |
|---|---|---|---|
| | 2 | | The system stores results |
| **Exceptions:** | - | | |
| **Include:** | - | | |

Table 6

| | |
|---|---|
| **Use Case UML ID:** | PR12 |
| **Use Case UML Name:** | Annotate re source |
| **Actors:** | |
| **Description:** | A resource (e.g., a document) is associated with an annotation |
| **Trigger:** | A user of community, who wants to communicate information |
| **Preconditions:** | Document isn't already in check out |
| **Postconditions:** | A document with annotations |
| **Normal Flow:** | Point 1 of Use Case UML, labeled Collaborative |
| **Exceptions:** | |
| **Include:** | |

| | Actor Input | System Response |
|---|---|---|
| 1 | The user checks a document out | |
| 2 | The user assigns a annotation to a resources | |
| 3 | | The system stores the annotation |
| 4 | The user checks a document of point 1 in | |

Table 12

# 8 Appendix

The following sections describe the infrastructure architecture that generated the requirements described in this document, the description of some proposals of collaborative applications, some recommended practices for requirement analysis template, guidelines for use case design, and general guidelines for developing user interfaces.

## 8.1 Evaluation infrastructure architecture requirements

Within an evaluation framework such as CLEF or the Promise platform, many people are involved in different tasks, such as organizing, creating topics, managing collections and handling participants and submission, and choosing measures and running the final evaluations. In some of these stages of the evaluation process, there is the need for collaborative information handling procedures. In order to identify the requirements for collaboration, we investigated three different domains (see Section 1.2).

This Section presents a list of identified needs for collaborative activities that may occur within an evaluation platform such as in Promise.

### 8.1.1 Actors

The basic set of actors within a Promise framework has been defined in work package 3, task 3.1. This set of actors comprise:

- organizers
- participants
- relevance assessors
- topic creators
- site administrators
- other researchers
- annotators.

Each of these actors is described along a set of activities, or tasks. These are further broken down in more detailed activities, or sub-tasks.

### 8.1.2 Collaborative information handling requirement elicitation

The list of identified steps has been enhanced through extensive communication with key persons of the main organizers of CLEF tracks representing three different domains:

- the patent domain, represented by IRF,
- the radiology domain, represented by HES-SO, and

- the Cultural heritage domain, represented by UBER .

Based on these communications, different needs for collaboration were reported and the list of actors and the activities performed by each category of actor have been enhanced.

### 8.1.3 Requirements for collaborative information handling in Promise.

This section presents the full list of actors and tasks performed within each category of actors. Furthermore, each task has been assigned one or several sub-tasks. In the case that new tasks or sub-tasks have been inserted diverging from the original set of actors, tasks and sub-tasks, we have added a (new) label. However, some of these tasks and sub-tasks are not valid for all domains. Certain domains may have different characteristics than other domains.

After each category of actors, a summary of identified and requested support for collaborative information handling is presented. The summary has two levels: the identified needs and the suggested support for collaboration.

#### *8.1.3.1 Categories of actors and evaluation tasks*

**Organizers**

- preparation of data
- copyright forms
- data on which tasks are available in a specific year
- metadata forms to collect metadata
- add data
- information about the specific track
- tasks and collections
- view collection data
- collection statistics
- documents
- metadata
- language distributions
- citations
- classifications
- annotations
- images
- cases description
- data usage
- create topics/information needs (cf. Topic creator)
- access to collection
- create information need / topic

- need to have access on user data through survey etc.
- view user data
- login/password
- copyright forms signed
- registrations, submissions
- data supplied
- handle submissions:
    - define accepted formats
    - check submissions
    - format (automatically if possible)
    - completeness
    - content
- prepare for evaluation
- run evaluations:
    - choose measures (usually trec_eval with a number of potential measures
    - define evaluation chains
    - choose qrels
    - run files
    - significance tests
- gather and combine the results
- visualization of the track results (centralized)
- upload results
- make results available to participants in the form of the qrels and the lists with the ranked results of participants.

**What an organizer needs**

There is an extensive collaboration among the CLEF track organizers during the complete evaluation campaign. On the organizers level, the following has been pointed out as important requirements for collaborative activities:

- The procedures and standards of preparation of data normally only involve the organizers; however it is communicated to other track organizers.
- During the actual track performance the organizers initiate collaboration with
    - participants,
    - assessors,
    - topic creators, and
    - visitors (other researchers),

in order to discuss and prepare performance measures, statistical analysis, and scientific papers.

This call for introducing a common and collaborative discussion area/platform in which discussions could take place and documents could be shared and where all user groups can collaborate with the organizers or, if needed, with other users. This platform should be able to support for collaborative information handling tasks across both a horizontal and vertical levels:

- it allows the organizers to collaborate with the other users,
- within a specific track as well as,
- among other organizers in other tracks.
- it provides tools that allow organizers to define and prepare relevant measures for a task and communicate these measures to other actors, such as participants, assessors, topic creators, and visitors.
- it provides tools that enable statistical analysis of data in a collaborative way. For example, single participants could compare data with other participants and organizers.

The organizers should be able to view results in different ways and at different levels. It should support collaborative information handling actions for the different people involved an evaluation platform such as: organizers, content providers, track participants, and system builders.

The system should keep the data consistent, limiting the amount of manual work and emails exchanges, by:

- Introducing a common working area that allows the organizer to view, in a centralized view, the results of the evaluation. This may involve specific results or comparing results from several participants. Tools that adapt to the type of data needed to be designed.
- Furthermore, different levels of statistics could be shared and collaborative tools could then allow for cooperative manipulation of these statistics. For example: data characteristics (based on metadata), data usage and experiments (e.g. number per lab, per task, etc.). Different views of these statistics could be displayed.

Also at the participant level, there may be a requirement of having a communication tool (interface) between groups of participants and experts on the organizational level. Amongst others the participants wanted more assistance for newcomers,

A Q&A tool may be used to solve this requirement.

It would be really helpful to be able of making different components of a retrieval system available between researchers and participants. Such components can be combined in different ways and everyone can concentrate on their main objective.

Support for collaboration:
A tool that a) handle communication for this exchange or discussion on sharing components of retrieval systems or b) a way of pointing to components that are free for use or a common area/storage area in which these components could be offered to others.

**Participant**

- registering
- download of data
    - data collection
    - topics
    - guidelines
- analysis
    - analyze data, indices, and databases
    - analyze topics, create queries
- IR system set-up
- submission
    - submit runs
    - submit metadata
    - submit results
    - submit results other than runs and metadata
- run evaluations of own additional results
    - together with significance tests
- receive measurements and validation based on the experiments,
    - interpretation of results
- compare to other participants
    - visualization of own results
- view user account
    - own submitted runs
    - other participants

**What a participant needs**

The following kinds of collaborative information handling activities are envisioned at the participant level:

There is an expressed need for collaboration between a) a participant and other participants and b) between an organizer and a participant as regards to feedback processes and means for different types of comparisons.

Support for collaboration is needed, especially between organizers and participants and among participants. A platform or a common work area would provide for:

- direct feedback,
- data comparison, and
- data analysis comparison.

There is an expressed need for sharing evaluation components and intermediate results between participants and organizers and among participant. Examples of this could be the sharing of standard sets of resources for result comparison and the possibility of sharing resources such as corpora, stop words list etc. It could also mean that collaboration may be required if and when participants have loosely integrated systems. All participants usually use different techniques for text retrieval and/or visual retrieval and/or different combination of methods. This would be a challenging field for collaboration and exchange. Collaboration between participants involving that a text retrieval researcher can exchange tools with an image retrieval researcher in order to be able to combine results and build better system.

A common work area may have a specific folder/area in which specific evaluation components and sets of test results can be viewed by other participants or/and organizers.

Also at the participant level, there may be a requirement of having a communication tool (interface) between groups of participants and experts on the organizational level.

There is a need for collaboration in session-based retrieval. Such an example rises when a participant performs an image/chemical structure search adding a textual search (e.g., in step one: system 1 provides an output. In step 2, the participant takes this output as an (additional) input while searching system 2.

Within some domains there are one or more sub-tasks, for example on medical retrieval, Wikipedia retrieval, photo annotation etc. In these situations collaboration only within a subtask really would make sense.

**Relevance assessor**

- access information needs
    - topics to be judged
    - documents to be judged
    - citations and classifications in them that have to be judged for relevance

        - languages
            - the level of language skills of the assessors
            - cross-lingual features
        - description for each of the relevance categories to have consistent judgments
    - sign relevance
        - grade
        - hierarchy of assessors

**What a relevance assessor needs**

The following kinds of collaborative information handling activities are envisioned at the assessor level.

From the point of view of an interaction among assessors, there is an expressed collaborative and communication need during the joint relevance assessment, in order to discuss the topics, to get help with difficult document to assess, and to manage retrieved results.

This joint collaboration could be solved though a tool that allow for (anonymous) communication among assessors.

During the relevance assessments, the assessors want to add/edit collaborative notes in order to share knowledge and experience. This may also include a requirement saying that the relevance assessors need to have access to topics in order to be able to improve the data. That calls for including in the work area an annotation tool for the assessors. Common information work areas are required as well, involving tools for visualization, annotations and data improvement.

Assessors also want to keep decision logs including the modifications leading to a final decision. For example, relevance assessors with different language skills may be able to access topics and other documents in order to review the relevance decisions.

The system allows for extracting the complete log for a relevance assessment or merging the history for a specific topic and its related documents.

An overall proposal for collaborative information handling activity is to view joint evaluation activities, which means that it is needed to pinpoint the specific stages on which this collaboration will rest upon.

**Topic creator**

- access and view topic collection
- create topics/information needs

**What a topic creator needs**

Topic creator needs to comment and discuss on topics in different languages

A cross-lingual problem is the translation of a topic in several languages. To address this need, the system should provide the users with a work area for topic creators equipped with a text and/or voice device for synchronous or asynchronous communication. This tool could also be connected to the requirements described below.

Topic creators need to annotate the document relevance assessments.

A tool like this would allow the actors to make annotations and to see other annotations made on the same topic.

**Site administrator**

- controls access rights to data collections
    - for various types of users (participants, assessors, admin, other)
    - through licensing
    - browsing data
    - visualizing data
    - using a tool for data analysis
    - defining test sub-collections with specific criteria
- controls access rights to results
    - for various types of users (participants, assessors, admin, other)
        - though notifications

**What a site administrator needs**

No specific requirement reported. See organizer.

**Other “researcher”**

- (using accumulated campaign data for research/evaluation purposes outside the context of an evaluation campaign),
- browse the data repositories
    - tracks and track history

    - topics
    - results
    - statistics
    - documentation
    - visualization
- not allowed to download large amounts of data (at least licensed)?
- not allowed to upload experiments?
- not able to modify anything on the promise infrastructure?
- logs tracing the activity created.

**What a researcher needs**

A researcher outside the evaluation campaign need to view results, data, topics, and statistics from previous campaigns in order to get an overview on what has previously been done and how data from different campaigns have been treated.

**Annotator**

- access collection
- images
- assign annotation
- validate annotations
- reach an agreement

**What an annotator needs**

Within a task such as to handle images, assigning annotations are being made by different people. There may be a need for consensus or validation. Annotators want to collaborate while making annotations.

Among annotators: A work area that disposes of a tool allowing for (anonymous) inter-annotator agreements among annotators. This could be used on one or several evaluation measures as a scaling factor (viewing a low or high level of agreement of the annotators for a concept). The collaborative tool could enable, for example, that 3 persons can look at all images in an image task and annotate them with concepts.

An organizer could also use the tool in order to manage, for example, invalid or inconsistent annotation. At the same time when annotations are made, wrong annotations could be removed.

## 8.2 Collaborative software

The design intent of collaborative software is to enable more effective team collaboration; to this aim it is useful to understand the three primary ways in which humans interact: conversations, transactions, and collaborations.

*Conversational interaction* is an exchange of information between two or more persons where the primary purpose of the interaction is discovery or relationship building. There is no central entity around which the interaction resolves, but is a free exchange of information with no defined constraints. Communication technology such as telephones, instant messaging, and e-mail are generally enough for conversational interactions.

*Transactional interaction* involves the exchange of transaction entities where a major function of the transaction entity is to alter the relationship among participants. The transaction entity is in a relatively stable form and constrains or defines the new relationship: e.g., one participant exchanges money for goods and becomes a customer. Transactional interactions are most effectively handled by transactional systems that manage state and commit records from persistent storage.

In *collaborative interactions* the main function of the participant's relationship is to alter a collaboration entity (i.e., the converse of transactional). The collaboration entity is not as formal as the transactional case. Examples include the development of an idea, the creation of a design, and the achievement of a shared goal. Therefore, real collaboration technologies deliver the functionality for many participants to argument a shared object (e.g., a document). Record or document management, threaded discussions, audit, history, annotations, and other mechanisms designed to capture the efforts of many into a managed content environment are typical technologies adopted in this context.

Collaboration cannot take place in a vacuum: it requires individuals working together in a coordinated fashion, towards a common goal. Accomplishing the goal is the primary purpose for bringing the team together. Collaborative software helps the action-oriented team to work together over geographic distances by providing tools that facilitate communication, collaboration, and the process of problem solving by providing the team with a common means for communicating ideas. Collaborative software should support the individuals that make up the team and the interactions between them during the group decision making process. Today's teams are composed of members from around the globe with many using their second or third language in communicating with the group. This provides cultural as well as linguistic challenges for any software that support ancillary systems, such as budgets and physical resources.

Brainstorming is considered to be a tenant of collaboration, with the rapid exchange of ideas facilitating the group decision making process. Collaborative software provides areas that support multi-user editing with virtual whiteboards and chat or other forms of communication. Better solutions record the process and provide revision history. A collaboration platform is a unified electronic platform that supports synchronous and asynchronous communication through a variety of devices and channels.

An extension of this is the *collaborative media*, a kind of software that allows several concurrent users to create and manage information in a website. Some sites with publicly accessible content based on collaborative software are: WikiWikiWeb, Wikipedia, and Everything2. According to the adopted method we can classify them into: Web-based collaborative tools, and software collaborative tools.

### 8.2.1 Collaborative working environment

A collaborative working environment (CWE) supports people in their individual and cooperative work. Research in CWE involves organizational, technical, and social issues.

The following applications or services are considered key elements of a CWE:

- E-mail
- Instant messaging
- Application sharing
- Videoconferencing
- Collaborative workspace and document management
- Task and workflow-management
- Wiki group or community effort to edit wiki pages (e.g. wiki pages describing concepts to enable a common understanding within a group or community)
- Blogging where entities are categorized by groups or communities or other concepts supporting collaboration
- Annotation
- Feedback

Working practices are evolving from the traditional proximity or geographical collocation paradigm to a virtual collocation paradigm where professionals work together whatever their geographical location. In this context, e-professionals use a collaborative working environment which provides the capabilities to share information and exchange views in order to reach a common understanding. Such a level of common understanding enables an effective and efficient collaboration among different experts.

The concept of CWE is derived from concept of virtual workspaces, and is related to the concept of e-work; it extends the traditional concept of the professional to group of professionals conducts their collaborative work through the use of collaborative working environments.

The CWE concept refers to online collaboration (i.e. virtual teams, mass collaboration, and massively distributed collaboration); online communities of practice, such as the open source community, and open innovation principles.

### 8.2.2 Annotation systems

With an annotation system a user can add, modify, or remove information from a resource without modifying the resource itself. Annotations can be thought as a layer on top of the existing resource, and this annotation layer is usually visible to other users who share the same annotation system. In such cases, the annotation system is a type of “social software”.

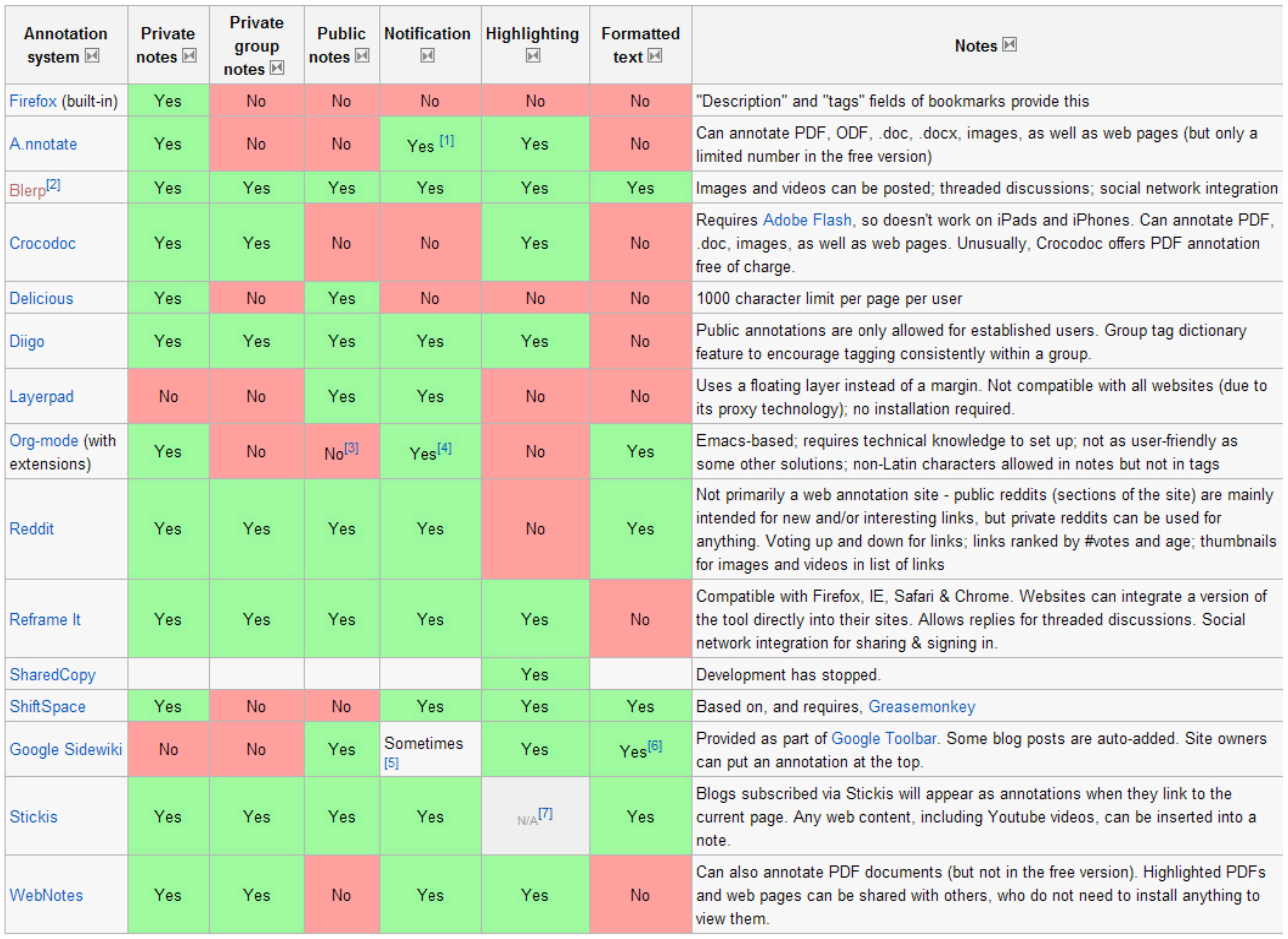
#### *8.2.2.1 Comparison of web annotation systems*

There are many annotation systems that often require additional software provided by a third party. A description of all relevant proposals is out of the scope of this document; Figure 3 describes the most relevant ones, together with their characteristics.

#### *8.2.2.2 The Annotea Project*

Annotea is a W3C project. Annotea enhances collaboration via shared metadata based Web annotations, bookmarks, and their combinations. By annotations we mean comments, notes, explanations, or other types of external remarks that can be attached to any Web document or a selected part of the document without actually needing to touch the document. When the user gets the document he can also load the annotations attached to it from a selected annotation server or several servers and see what his peer group thinks. Similarly shared bookmarks can be attached to Web documents to help organize them under different topics, to easily find them later, to help find related material and to collaboratively filter bookmarked material.

Annotea is an open architecture; it uses and helps to advance W3C standards when possible. For instance, it uses a Resource Description Framework based annotation schema for describing annotations as metadata and XML for locating the annotations in the annotated document. Similarly, a bookmark schema describes the bookmark and topic metadata.



| Annotation system | Private notes | Private group notes | Public notes | Notification | Highlighting | Formatted text | Notes |
|---|---|---|---|---|---|---|---|
| Firefox (built-in) | Yes | No | No | No | No | No | "Description" and "tags" fields of bookmarks provide this |
| A.nnotate | Yes | No | No | Yes [1] | Yes | No | Can annotate PDF, ODF, .doc, .docx, images, as well as web pages (but only a limited number in the free version) |
| Blerp[2] | Yes | Yes | Yes | Yes | Yes | Yes | Images and videos can be posted; threaded discussions; social network integration |
| Crocodoc | Yes | Yes | No | No | Yes | No | Requires Adobe Flash, so doesn't work on iPads and iPhones. Can annotate PDF, .doc, images, as well as web pages. Unusually, Crocodoc offers PDF annotation free of charge. |
| Delicious | Yes | No | Yes | No | No | No | 1000 character limit per page per user |
| Diigo | Yes | Yes | Yes | Yes | Yes | No | Public annotations are only allowed for established users. Group tag dictionary feature to encourage tagging consistently within a group. |
| Layerpad | No | No | Yes | Yes | No | No | Uses a floating layer instead of a margin. Not compatible with all websites (due to its proxy technology); no installation required. |
| Org-mode (with extensions) | Yes | No | No[3] | Yes[4] | No | Yes | Emacs-based; requires technical knowledge to set up; not as user-friendly as some other solutions; non-Latin characters allowed in notes but not in tags |
| Reddit | Yes | Yes | Yes | Yes | No | Yes | Not primarily a web annotation site - public reddits (sections of the site) are mainly intended for new and/or interesting links, but private reddits can be used for anything. Voting up and down for links; links ranked by #votes and age; thumbnails for images and videos in list of links |
| Reframe It | Yes | Yes | Yes | Yes | Yes | No | Compatible with Firefox, IE, Safari & Chrome. Websites can integrate a version of the tool directly into their sites. Allows replies for threaded discussions. Social network integration for sharing & signing in. |
| SharedCopy | | | | | Yes | | Development has stopped. |
| ShiftSpace | Yes | No | No | Yes | Yes | Yes | Based on, and requires, Greasemonkey |
| Google Sidewiki | No | No | Yes | Sometimes [5] | Yes | Yes[6] | Provided as part of Google Toolbar. Some blog posts are auto-added. Site owners can put an annotation at the top. |
| Stickis | Yes | Yes | Yes | Yes | N/A[7] | Yes | Blogs subscribed via Stickis will appear as annotations when they link to the current page. Any web content, including Youtube videos, can be inserted into a note. |
| WebNotes | Yes | Yes | No | Yes | Yes | No | Can also annotate PDF documents (but not in the free version). Highlighted PDFs and web pages can be shared with others, who do not need to install anything to view them. |

**Figure 3: Comparison of web annotations toolkits**

Annotea is part of Semantic Web effort. It provides a Resource Description Framework metadata based extendible framework for rich communication about Web pages while offering a simple annotation and bookmark user interface. The annotation metadata can be stored locally or in one or more annotation servers and presented to the user by a client capable of understanding this metadata and capable of interacting with an annotation server with the HTTP service protocol.

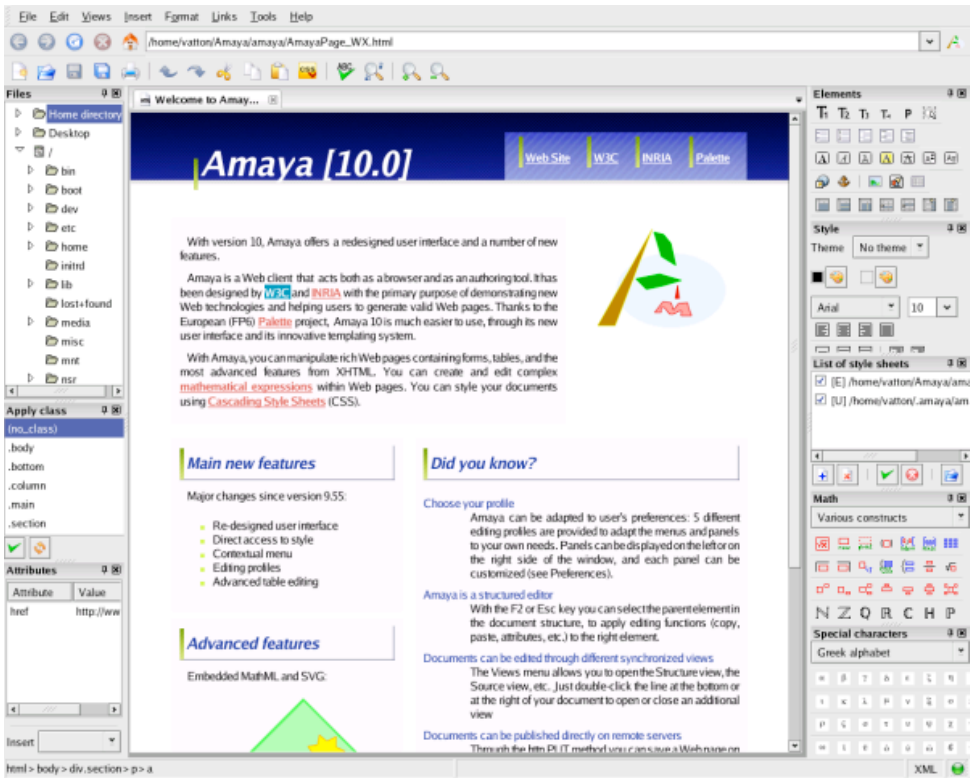
The first client implementation of Annotea is the W3C's Amaya editor/browser.

#### *8.2.2.3 Amaya*

Amaya is a Web editor, i.e., a tool used to create and update documents directly on the Web. Browsing features are seamlessly integrated with the editing and remote access features in a uniform environment. This follows the original vision of the Web as a space for collaboration and not just a one-way publishing medium.

Work on Amaya started at W3XC in 1996 to showcase Web technologies in a fully-featured Web client. The main motivation for developing Amaya was to provide a framework that can integrate as many W3C technologies as possible. It is used to demonstrate these technologies in action while taking advantage of their combination in a single, consistent environment.

Amaya includes a collaborative annotation application based on Resource Description Framework (RDF).



Amaya [10.0]

With version 10, Amaya offers a redesigned user interface and a number of new features.

Amaya is a Web client that acts both as a browser and as an authoring tool. It has been designed by W3C and INRIA with the primary purpose of demonstrating new Web technologies and helping users to generate valid Web pages. Thanks to the European (FP6) Palette project, Amaya 10 is much easier to use, through its new user interface and its innovative templating system.

With Amaya, you can manipulate rich Web pages containing forms, tables, and the most advanced features from XHTML. You can create and edit complex mathematical expressions within Web pages. You can style your documents using Cascading Style Sheets (CSS).

Main new features

Major changes since version 9.55:

- Re-designed user interface
- Direct access to style
- Contextual menu
- Editing profiles
- Advanced table editing

Advanced features

Embedded MathML and SVG:

Did you know?

Choose your profile
Amaya can be adapted to user's preferences: 5 different editing profiles are provided to adapt the menus and panels to your own needs. Panels can be displayed on the left or on the right side of the window, and each panel can be customized (see Preferences).

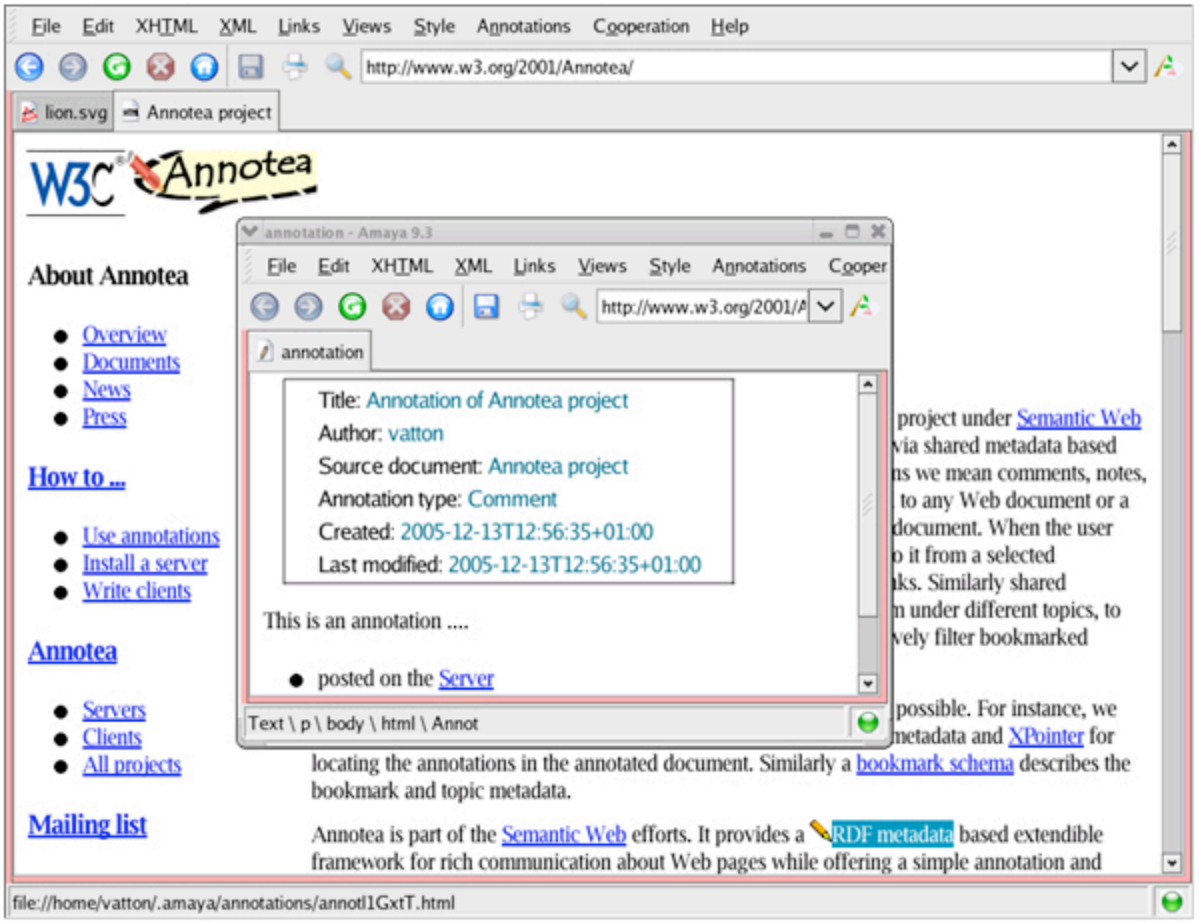
Amaya is a structured editor
With the F2 or Esc key you can select the parent element in the document structure, to apply editing functions (copy, paste, attributes, etc.) to the right element.

Documents can be edited through different synchronized views
The Views menu allows you to open the Structure view, the Source view, etc. Just double-click the line at the bottom or at the right of your document to open or close an additional view

Documents can be published directly on remote servers

html > body > div.section > p > a

**Figure 4: An Amaya screenshot**



Title: Annotation of Annotea project
Author: vatton
Source document: Annotea project
Annotation type: Comment
Created: 2005-12-13T12:56:35+01:00
Last modified: 2005-12-13T12:56:35+01:00

This is an annotation ....

- posted on the Server

project under Semantic Web
via shared metadata based
ns we mean comments, notes,
to any Web document or a
document. When the user
o it from a selected
ks. Similarly shared
n under different topics, to
vely filter bookmarked

possible. For instance, we
metadata and XPointer for
locating the annotations in the annotated document. Similarly a bookmark schema describes the bookmark and topic metadata.

Annotea is part of the Semantic Web efforts. It provides a RDF metadata based extendible framework for rich communication about Web pages while offering a simple annotation and

**Figure 5: An Annotea screenshot**

## 8.3 IEEE 830-1998 Recommended Practice Software Requirements Specifications

In designing computer-based information systems, special attention must be given to software supporting the user interface. To offer some general requirements for user interface, we will use a simplified version of the IEEE 830-1998 Recommended Practice Software Requirements Specifications guidelines.

This is a recommended practice for writing software requirements specifications. It describes the content and qualities of a good software requirements specification (SRS).

This recommended practice does not identify any specific method, nomenclature, or tool for preparing an SRS.

Software requirements will be written according to the following structure:

1. Introduction
    1.1. Purpose
    1.2. Scope
    1.3. Definitions, acronyms, and abbreviations
    1.4. References
    1.5. Overview
2. Overall description
    2.1. Product perspective
    2.2. Product functions
    2.3. User characteristics
    2.4. Constraints
    2.5. Assumptions and dependence
3. Specific requirements
    3.1. External interface requirements
        3.1.1. User interfaces
        3.1.2. Software interfaces
        3.1.3. Communications interfaces
    3.2. Functional requirements
        3.2.1. Mode 1
            3.2.1.1. Functional requirement 1.1
            .
            .
            .
            3.2.1.2. Functional requirement 1.*n*
        3.2.2. Mode 2

## 1. Introduction

The introduction of the SRS should provide an overview of the entire SRS. It should contain the following subsections:

### 1.1. Purpose

This subsection should delineate the purpose of the SRS

### 1.2 Scope

This subsection should

a) Identify the software product(s) to be produced by name (e.g., Host DBMS, Report Generator, etc.);
b) Explain what the software product(s) will, and, if necessary will not do
c) Be consistent with similar statements in higher-level specifications (e.g., the system requirements specification), if they exist.

### 1.3. Definitions, acronyms, and abbreviations

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

### 1.4. References

This subsection should

a) Provide a complete list of all documents referenced elsewhere in the SRS
b) Identify each document by title, report number (if applicable), date, and publishing organization
c) Specify the sources from which the references can be obtained

### 1.5. Overview

This subsection should

a) Described what the rest of the SRS contains
b) Explain how the SRS is organized

## 2. Overall description

This section of the SRS should describe the general factors that affect the product and its requirements. This section does not state specific requirements.

### 2.1. Productive perspective

This subsection of the SRS should put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS define a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful.

This subsection should also describe how the software operates inside various constraints. For example, these constraints could include

a) System interfaces
b) User interfaces
c) Software interfaces
d) Communications interfaces
e) Operations
f) Site adaptation requirements.

#### 2.1.1. System interfaces

This should list each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system.

#### 2.1.2. User interfaces

This should specify the following:

a) *The logic characteristics of each interface between the software product and its users*. This includes those configuration characteristics (e.g., required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys) necessary to accomplish the software requirements
b) *All the aspects of optimizing the interface with the person who must use the system*. This may simply comprise a list of do's and don'ts on how the system will appear to the user. One example may be a requirement for the option of long or short error messages. Like all others, these requirements should be verifiable, e.g., "a clerk typist grade 4 can do function X in Z min after 1 h of training" rather than "a typist can do function X".

#### 2.1.3. Software interfaces

This should specify the use of the other required software product (e.g., a data management system, an operating system, or a mathematical package), and interfaces with other application systems (e.g., the linkage between an accounts receivable system and a general ledger system). For each required software product, the following should be provided:

a) Name
b) Mnemonic
c) Specification number
d) Version number
e) Source
f) Site adaptation requirements

For each interface, the following should be provided:

a) Discussion of the purpose of the interfacing software as related to this software product.
b) Definition of the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.

#### 2.1.4. Communiations interfaces

This should specify the various interfaces to communications such as local network protocols, etc.

#### 2.1.5. Operations

This should specify the normal and special operations required by the user such as

a) The various modes of operations in the user organization (e.g., user-initiated operations)
b) Periods of interactive operations and periods of unattended operations
c) Data processing support functions
d) Backup and recovery operations

#### 2.1.6. Site adaptation requirments

This should

a) Define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode (e.g., grid values, safety limits, etc.)
b) Specify the site or mission-related features that should be modified to adapt the software to a particular installation

### 2.2. Product functions

This subsection of the SRS should provide a summary of the major functions that the software will perform. For example, an SRS for a counting program may use this part to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those function requires.

Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (if one exists) that allocates particular functions to the software product. Note that for the sake of clarity

a) The functions should be organized in a way that makes the list of functions understandable to the customer or to anyone else reading the document for the first time
b) Textual or graphical methods can be used to show the different functions and their relationships. Such a diagram Is not intended to show a design of a product, but simply shows the logical relationships among variables

### 2.3. User characteristics

This subsection of the SRS should describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise.

### 2.4. Constraints

This subsection of the SRS should provide a general description of any other items that will limit the developer's options. These include

a) Regulatory policies

b) Interfaces to other applications
c) Parallel operation
d) Control functions
e) Reliability requirements
f) Criticality of the application

### 2.5. Assumptions and dependencies

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

## 3. Specific requirements

This sector of the SRS should contain all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system, and a functions performed by the system in response to an input or in support of an output. As this is often the largest and most important part of the SRS, the following principles apply:

a) Specific requirements should be cross-references to earlier documents that relate
b) All requirements should be uniquely identifiable
c) Careful attention should be given to organizing the requirements to maximize readability

Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements

### 3.1. External interfaces

This should be a detailed description of all inputs into and outputs from the software system. I should complement the interface descriptions and should not repeat information there.

It should include both content and format as follows:

a) Name of item

b) Description of purpose
c) Source of input or destination of output
d) Valid range, accuracy, and/or tolerance
e) Units of measure
f) Timing
g) Relationships to other inputs/outputs
h) Screen formats/organization
i) Window formats/organization
j) Data formats
k) Command formats
l) End messages

### 3.2. Functional requirements

Functional requirements should define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as "shall" statements starting with "The system shall…"

These include

a) Validity checks on the inputs
b) Exact sequence of operations
c) Responses to abnormal situations, including
   a. Overflow
   b. Communication facilities
   c. Error handling and recovery
d) Effect of parameters
e) Relationship of outputs to inputs, including
   a. Input/output sequences
   b. Formulas for input to output conversion

### 3.3. Performance requirements

This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include the following

a) The number of terminals to be supported
b) The number of simultaneous users to be supported
c) Amount and type of information to be handled

## 8.4 UML Use Cases design guidelines

UML Use cases describe system behavior from an actor's point of view as scenario-driven threads through the functional requirements.

Within systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals.

"Each use case focuses on describing how to achieve a goal or a task. For most software projects, this means that multiple, perhaps dozens of use cases are needed to define the scope of the new system. The degree of formality of a particular software project and the stage of the project will influence the level of detail required in each use case."

Use cases should not be confused with the features of the system. One or more features (a.k.a. "system requirements") describe the functionality needed to meet a stakeholder request or user need (a.k.a. "user requirement"). Each feature can be analyzed into one or more use cases, which detail cases where an actor uses the system. Each use case should be traceable to its originating feature, which in turn should be traceable to its originating stakeholder/user request.

Use cases treat the system as a black box, and the interactions with the system, including system responses, are perceived as from outside the system. This is a deliberate policy, because it forces the author to focus on what the system must do, not how it is to be done, and avoids making assumptions about how the functionality will be accomplished.

A use case should:

- Describe what the system shall do for the actor to achieve a particular goal.
- Include no implementation-specific language.
- Be at the appropriate level of detail.
- Not include detail regarding user interfaces and screens. This is done in user-interface design, which references the use case and its business rules.

### 8.4.1 History

In 1986, Ivar Jacobson, later an important contributor to both the Unified Modeling Language (UML) and the Rational Unified Process (RUP), first formulated the visual modeling technique for specifying use cases. Originally he used the terms usage scenarios and usage case, but found that neither of these terms sounded natural in English, and eventually he settled on the term use case. Since Jacobson originated use case modeling, many others have contributed to improving this technique, including Kurt Bittner, Ian Spence, Alistair Cockburn, Gunnar Overgaard, Karin Palmquist and Geri Schneider.

Network of Excellence co-funded by the 7th Framework Program of the European Commission, grant agreement no. 258191

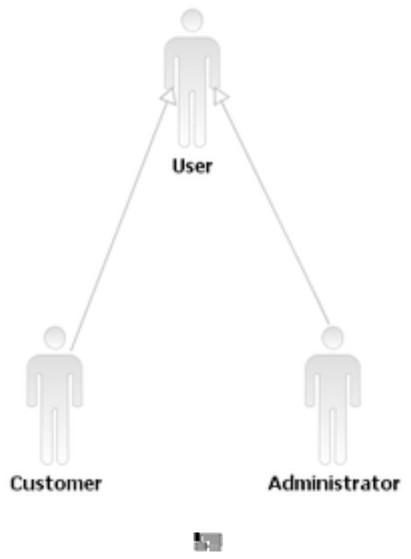
During the 1990s use cases became one of the most common practices for capturing functional requirements. This is especially the case within the object-oriented community where they originated, but their applicability is not restricted to object-oriented systems, because use cases are not object-oriented in nature.

### 8.4.2 Business vs. System Use Cases

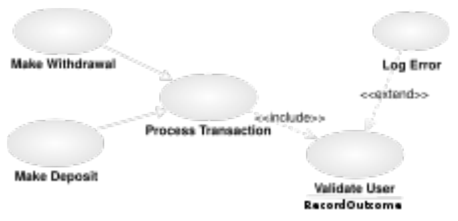
Use cases may be described at the abstract level (business use case, sometimes called essential use case), or at the system level (system use case). The differences between these are the scope.

- A **business use case** is described in technology-free terminology which treats system as a black box and describes the business process that is used by its business actors (people or systems external to the process) to achieve their goals (e.g., manual payment processing, expense report approval, manage corporate real estate). The business use case will describe a process that provides value to the business actor, and it describes *what* the process does. Business Process Mapping is another method for this level of business description.
- A **system use case** describes a system that automates a business use case or process. It is normally described at the system functionality level (for example, "create voucher") and specifies the function or the service that the system provides for the actor. The system use case details *what* the system will do in response to an actor's actions. For this reason it is recommended that system use case specification begin with a verb (e.g., *create* voucher, *select* payments, *exclude* payment, *cancel* voucher). An actor can be a human user or another system/subsystem interacting with the system being defined.

### 8.4.3 Diagram building blocks



Actor inheritance



Use case relationships

Interaction among actors is not shown on the use case diagram. If this interaction is essential to a coherent description of the desired behavior, perhaps the system or use case boundaries should be re-examined. Alternatively, interaction among actors can be part of the assumptions used in the use case.

- Use cases

  A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

- Actors

  An actor may be a person, a device, another system or sub-system, or time that plays a role in one or more interactions with the system. Actors represent the different roles that something outside has in its relationship with the system whose functional requirements are being specified. An individual in the real world can be represented by several actors if the individual plays several different roles and has multiple goals w.r.t. a system. These actors interact with the system and do some action on it.

- System boundary boxes (optional)

  A rectangle is drawn around the use cases, called the system boundary box, to denote the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not.

### 8.4.4 Actor Generalization

One popular relationship between Actors is Generalization/Specialization. This is useful in defining overlapping roles between actors. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general actor

### 8.4.5 Use Case Relationships

Four relationships among use cases are used often in practice.

#### 8.4.5.1 Include

In one form of interaction, a given use case may *include* another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case".

*The first use case often depends on the outcome of the included use case*. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»". This usage resembles a macro expansion where the included use case behavior is placed inline in the base use case behavior. There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behavior of another, you simply write *include* followed by the name of use case you want to include, as in the following flow for *track order*.

#### 8.4.5.2 Extend

In another form of interaction, a given use case (the extension) may *extend* another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". The notes or constraints may be associated with this relationship to illustrate the conditions under which this behavior will be executed.

Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case. Depending on the modeler's approach "optional" may mean "potentially not executed with the base use case" or it may mean "not required to achieve the base use case goal".

#### 8.4.5.3 Generalization

In the third form of relationship among use cases, a *generalization/specialization* relationship exists. A given use case may have common behaviors, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation).

#### 8.4.5.4 Associations

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case. The arrowheads imply control flow and should not be confused with data flow.

### 8.4.6 Use case templates

There is no standard template for documenting detailed use cases. A number of competing schemes exist, and individuals are encouraged to use templates that work for them or the project they are on. Standardization within each project is more important than the detail of a specific template. There is, however, considerable agreement about the core sections; beneath differing terminologies and orderings there is an underlying similarity between most use cases. Different templates often have additional sections, e.g., assumptions, exceptions, recommendations, technical requirements. In our case, you have chosen the following sections:

Use case id

> Id of the Use Case

Use case name

> A descriptive name provides a unique identifier for the use case. It should be written in verb-noun format (e.g., *Borrow Books*, *Withdraw Cash*), it should describe an achievable goal (e.g., *Register User* is better than *Registering User*) and it should be enough for the end user to understand what the use case is about.

Primary Actor

> He/She/it is someone or something outside the system that acts on the system.

Secondary Actor(s)

> He/She/it is someone or something outside the system that is acted on by the system.

Description

> A description is used to capture the essence of a use case before the main body is complete. It provides a quick overview, which is intended to save the reader from having to read the full contents of a use case to understand what the use case is about.

Preconditions

A *preconditions* section defines all the conditions that must be true (i.e., describes the state of the system) for the *trigger* (see below) to meaningfully cause the initiation of the use case. That is, if the system is not in the state described in the preconditions, the behavior of the use case is indeterminate. Note that the preconditions are *not* the same thing as the "trigger" (see below): the mere fact that the preconditions are met does NOT initiate the use case.

Postconditions

The *post-conditions* section describes what the change in state of the system will be after the use case completes. Post-conditions are guaranteed to be true when the use case ends.

Triggers

A 'triggers' section describes the event that causes the use case to be initiated. This event can be external, temporal or internal. If the trigger is not a simple true "event" (e.g., the customer presses a button), but instead "when a set of conditions are met", there will need to be a triggering process that continually (or periodically) runs to test whether the "trigger conditions" are met: the "triggering event" is a signal from the trigger process that the conditions are now met.

Normal Flow

At a minimum, each use case should convey a *primary scenario*, or typical course of events, also called “normal flow”, "basic flow", "happy flow" and “Happy path”. The main basic course of events is often conveyed as a set of usually numbered steps. For example:

1. The system prompts the user to log on,
2. The user enters his name and password,
3. The system verifies the logon information,
4. The system logs user on to system.

Alternative paths or Exceptions

Use cases may contain secondary paths or alternative scenarios, which are variations on the main theme. Each tested rule may lead to an alternative path and when there are many rules the permutation of paths increases rapidly, which can lead to very complex documents.

## 8.5 Guideline for User Interface developments

A heuristic evaluation is a discount usability inspection method that helps to identify usability problems in the user interface design. It specifically involves expert evaluators examining the interface and judging its compliance with recognized usability principles (the "heuristics"). The main goal of heuristic evaluation is to identify any problems associated with the design of user interfaces.

Conversely, in PROMISE we propose these heuristics to all partners as guidelines for the design and implementation. of user interfaces. Before to design and to develop any user interface, we suggest reading and following these heuristics.

Below the ten heuristics:

### Visibility of the system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

### Match between system and the real world

The system should speak the user language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

### User control and freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

### Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

### Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

### Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user not has to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

### Flexibility and efficiency of use

Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

### Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

### Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

### Help and documentation

Even through it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's talk, list concrete steps to be carried put, and not be too large.

## References

[Ackerman, M. and Malone, T. 1990] Answer Garden: A tool for Growing Organizational Memory. Proceedings of ACM Conference on Office Information Systems 1990 (pp. 31-39). Cambridge, Massachusetts, United States: ACM.

[Baldoni, R. 2008] Scalable Publish/Subscribes Infrastructures handouts http://www.dis.uniroma1.it/~baldoni/SDslide_pubsub_2008_SD.pdf

[Ehrlich, K. and Cash, T. 1994] Turning Information into Knowledge: Information Finding as a Collaborative Activity. Proceedings of the First Annual Conference on the Theory and Practice of Digital Libraries, June 19-21, 1994 (pp. 119-125). College Station, Texas, USA.

[Fidel, R., Bruce, H., Pejtersen, A. M., Dumais, S., Grudin, J., and Poltrick, S. 2000] Collaborative Information Retrieval (CIR). The New Review of Information Behaviour Research, 2002, pp. 235-247.

[Foster, J. 2006] Collaborative information seeking and retrieval. In: Annual Review of Information Science and Technology, Volume 40, 2006, 329–356.

[Haake, J. M., Wiil, U. K. and Nürnberg, P. J. 1999] Openness in shared hypermedia workspaces: The case for collaborative open hypermedia systems. SIGWEB Newsletter, 8(3), 1999, pp 33-45.

[Hansen, P. and Järvelin, K. 2000] Collaborative information retrieval in an information-intensive domain. Information Processing and Management (IPM), 41 (5), pp. 1101-1119. Sep-2005.

[IEEE 1998] Recommended Practice for Software Requirements Specifications, IEEE 830-1998, http://standards.ieee.org/findstds/standard/830-1998.html

[Karamuftuoglu, M. 1998] Collaborative Information Retrieval: Towards a Social Informatics View of IR Interaction. JASIS, 49(12), 1070-1080, 1998.

[McDonald, D. W., and Ackerman, M. S. 1998] Just talk to me: A field study of expertise location. In: Poltrock, S & Grudin, J. (Eds). Proceedings of the ACM CSCW'98 Conference on Computer Supported Cooperative Work (pp. 315-324). New York: ACM Press.

[Morris, M.R. and Horvitz, E. 2007] SearchTogether: An Interface for Collaborative Web Search. In: Shen, C., Jacob, R. and Balakrishnan, R. (Eds.), Proceedings of the 20th annual ACM symposium on User interface software and technology, Newport, Rhode Island, USA, October 7 - 10, 2007, 3 – 12.

[Molich, R., and Nielsen, J. 1990] Improving a human-computer dialogue, Communications of the ACM 33, 3 (March), 338-348.

[Molich, R., and Nielsen, J. 1990] Improving a human-computer dialogue, Communications of the ACM 33, 3 (March), 338-348.

[Nielsen, J., and Molich, R. 1990] Heuristic evaluation of user interfaces, proc. ACM CHI'90 Conf. (Seattle, WA, 1-5 April, 249-256.

[Nielsen, J. 1994a] Enhancing the explanatory power of usability heuristics. Proc. ACM CHI'94 Conf. (Boston, MA, April 24-28), 152-158.

[Nielsen, J. 1994b] Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), usability Inspection Methods, John Wiley & Sons, New York, NY.

[O'Day, V. and Jeffries, R. 1993] Orienteering in an information landscape: how information seekers get from here to there. Proceedings of the conference on Human factors in computing systems. INTERCHI '93, (pp. 438-445), ACM, 1993, Amsterdam, The Netherlands, New York: ACM Press.

[Reddy, M.C., and Spence, P.R. 2008] Collaborative information seeking: A field study of a multidisciplinary patient care team. Information Processing & Management 22(1), 242-255.

[Sameer, T. 2006] RESTful Web Services, http://www.oracle.com/technetwork/articles/javase/index-137171.html

[Tang, A., Tory, M., Po, Barry., Neumann, Petra and Carpendale, S. 2006] Collaborative Coupling over Tabletop Displays - Proceedings of the SIGCHI 2006 Conference on Human Factors in computing systems, April 22–27, 2006, Montréal, Québec, Canada, 1181 – 1190.

[W3C 2005] Annotea Project, web site http://www.w3.org/2001/Annotea/

[W3C 2010] Amaya Project, web site http://www.w3.org/Amaya/