CIRLab: A Groupware Framework for Collaborative Information Retrieval Research

Juan M. Fernández-Luna, Juan F. Huete

Departamento de Ciencias de la Computación e Inteligencia Artificial. E.T.S.I. Informática y de Telecomunicación. Universidad de Granada, 18071, Granada, Spain

Ramiro Pérez-Vázquez

Departamento de Ciencias de la Computación. Facultad de Física, Matemática y Computación. Universidad Central de Las Villas, 50300, Santa Clara, Cuba

Julio C. Rodríguez-Cano

Departamento de Informática. Facultad de Informática y Matemática. Universidad de Holguín, 80100, Holguín, Cuba

Abstract

In this paper we describe the design of a groupware framework, CIRLab, for experimenting with collaborative information retrieval (CIR) techniques in different search scenarios. This framework has been designed applying design patterns and an object-oriented middleware platform to maximize its re-usability and adaptability in new contexts with a minimum of programming efforts. Our collaborative search application comprises three main modules: the Core, which supports various modern state-of-the-art CIR techniques that can be reused or extended in a distributed collaborative environment; the Facades Mediator, an event-driven notification service which allows easy integration between the Core and front-end applications; and finally, the Actions Tracker, which allows researchers to perform experiments on the different elements involved in the collaborative search sessions. The applying of this framework is illustrated through the analysis of the collaborative search-driven development case study.

Key words: Collaborative information retrieval, Collaborative search, Source code search, Search driven development, Framework, Groupware

Preprint submitted to Information Processing & Management November 13, 2009
1. Introduction

In the last few years, with the large amount of digital information, the demand for efficient approaches to better information access has increased. On the other hand, with the advances in distributed technologies and computer hardware, the remotely teamwork is becoming more frequently used. Consequently, some researchers have realized that collaboration is an important feature which should be analyzed in detail in order to be integrated with professional information retrieval (IR) systems, upgrading these to CIR systems.

An early definition of CIR was given by S. Dumais et al. in [4] as “any activity that collectively resolves an information problem taken by members of a workteam”. P. Hansen and K. Järvelin [19], other two researchers who considered collaboration as an important component in the IR process, defined CIR as “an information access activity related to specific problem solving activity that, implicitly or explicitly, involves human beings interacting with other human(s) directly and/or through texts (e.g., documents, notes, figures) as information sources in a work task related information search and retrieval process either in a specific workplace setting or in a more open community or environment”. C. Foley defines in [7] synchronous CIR systems as “a field concerned with the realtime, explicit collaboration which occurs when multiple users search together to satisfy a shared information need. This collaboration can take place either with the users working remotely, or, in a co-located setting”.

More recently, the community has adopted CIR as an emerging research field, which includes the research areas that traditionally have been part of IR, with a special emphasis on synchronous or asynchronous but explicit collaboration among people with shared information need in a co-located or distributed environment. A good starting point to develop CIR is the mixture of IR with other disciplines such as Human-Computer Interaction (HCI) and Computer-Supported Cooperative Work (CSCW) [19, 9, 14] as shown in Figure 1.
HCI and IR disciplines have both developed innovative techniques to address the challenge of navigating the complex information spaces, but so far their insights often failed to cross disciplinary borders. Human-Computer Interaction Information Retrieval (HCIR) has emerged in academic research and industry practice as the study of IR techniques that bring human intelligence into the searching process. This field brings together research in the fields of IR and HCI in order to create new kinds of searching systems that depend on continuous human control of the searching process [25]. In addition, CSCW can blend with IR and HCI in order to show CIR as a new interesting research area. CSCW is defined in [10, 34] as an activity carried out by a group of persons, but coordinated and assisted by computers.

Most of the CIR systems developed recently include some common features: session persistence, division of labor, knowledge sharing and awareness inherited from the different disciplines involved in CIR. For more evidence on the community’s interest to improve these new modern IR approaches, please read CIR\(^1\), HCIR\(^2\) and CIB\(^3\) proceedings.

There are a good number of CIR prototypes and systems (see Section 3 where some examples and their features are described), but the CIR community does not have ad-hoc toolkits, application program interfaces (APIs) or class libraries with public access that can be extended or reused. Therefore, the development of CIR applications or the experimentation with new research ideas requires great programming efforts. This motivates us to design CIRLab (Collaborative Information Retrieval Laboratory), as a groupware framework applying design patterns and an object-oriented middleware platform. CIRLab also wraps some open-source third party APIs (e.g. search engines). These features let us maximize the CIRLab’s reusability and adaptability in new CIR contexts.

In the CSCW context, the term groupware refers to an application that helps people work together collectively while located remotely (different place) or co-located (same place) from each other, and interacting synchronously (same time) or asynchronously (different time) [5]. One of the most general definitions was coined by Wells et al. in [39] “Groupware is the software and hardware for shared interactive environments”. A CIR system can support collaborative activities according to this two dimensions (place and time), but it necessarily has to support explicit collaboration. Finally, a CIR system can be defined as a groupware that supports explicit collaboration among group members in order to help them to satisfy their shared information need.

Our approach takes advantage of the synergistic relationship among design patterns, frameworks and middleware. Design patterns codify reusable design expertise that provides time-proven solutions to commonly occurring software problems that arise in particular contexts and domains. Frameworks provide

\(^1\)http://workshops.fxpal.com/jcdl2008/Default.aspx
\(^2\)http://research.microsoft.com/~ryenw/hcir2008/index.html
\(^3\)http://www.group2009conference.com/
both a reusable architecture, guided by patterns, and an integrated set of collaborating components that implement concrete realizations of the architecture. Middleware is reusable software that leverages patterns and frameworks to bridge the gap between the functional requirements of applications and the underlying operating systems, network protocol stacks, and data persistence [36].

The objective of this paper is to present the CIRLab design principles and characteristics, as well as to present a case study about how this can be instantiated with the collaborative search-driven development (CSDD) as a particular context, comparing and evaluating then different search scenarios. Search-driven development (SDD) is a new research area motivated by the observation that software developers spend most of their times in searching pertinent information that they need to solve their task at hand [2]. We identified that SDD context was a very interesting field where CIR features could be greatly exploited. For this reason we use the phrase CSDD to refer us to apply different CIR techniques in the SDD process.

CIRLab can be used both by developers unfamiliar with groupware development and IR techniques through its instantiation, as well as by experts in IR, HCI and CSCW fields interested in CIR research and experimentation.

We have organized this paper as follows. In the next section (2), we shall present an overview of patterns, frameworks, and middleware, describing how these technologies complement each other as CIRLab foundations. Then, in Section 3, we shall describe the main areas of research in CIR. After this, the design considerations of CIRLab in Section 4 are discussed. In addition, the illustration of the CIRLab instantiation through a case study of CSDD is described in Section 5, discussing an experimental evaluation as well. Section 7 concludes this paper exposing conclusions and the improvements foreseen for CIRLab.

2. Middleware, Frameworks and Patterns: CIRLab Foundations

The development of CIR prototypes and systems is a complex task, which involves software technologies in different areas such as distributed programming, information search and retrieval, collaboration among people, task coordination and many others. This situation is not ideal because it requires great programming efforts. Fortunately, design patterns, frameworks, and middleware are increasing their popularity since they have a high reusability impact and suitable relationships [36]. To address this CIR implementation problem, we have designed CIRLab as a groupware framework. Methodologies for the development of a framework have been suggested that use domain analysis (see Section 3 for a brief analysis of a CIR domain), software evolution, and design patterns. This section presents an overview of patterns, frameworks, and middleware and describes how these technologies complement each other to enhance their reuse and productivity.
2.1. Middleware

As mentioned before, a groupware supports collaboration among group members that can be in different places at the same time. This capability requires a distributed architecture, usually Web applications with a client/server architecture. This model is very useful when collaboration is asynchronous (e.g. in collaborative filtering or social networks), but in synchronous interactive situations as well as CIR systems it is not very efficient. For that reason we consider the use of a middleware-based architecture as mechanisms of communication among user clients, instead of a Web-based architecture. We wish to emphasize than with the middleware-based architecture it is also possible to surf the Web.

Over the past decade, a number of object-oriented middleware standards have emerged and matured, such as the Common Object Request Broker Architecture (CORBA) and Distributed Component Object Model (DCOM). Currently, the developers who are looking for an object-oriented middleware platform are offered some alternatives. Therefore, it is important to make careful selections. For example, .NET/WCF has the drawback that it supports only a limited number of languages and platforms. Java Remote Method Invocation (RMI) is a Java-only solution. CORBA has got the high degree of complexity of an aging platform, coupled with ongoing vendor attrition. Web Services have severe inefficiencies and the need of using proprietary development platforms, as well as security issues [21].

For our purposes, we have selected the Internet Communications Engine 4 (ICE) because its applications are open-source, suitable for use in heterogeneous environments: client and server can be written in different programming languages, run on different operating systems and machine architectures, and they communicate using a variety of networking technologies [21].

2.2. Frameworks

A framework is a collection of classes that provides a set of services for a particular domain; a framework thus exports a number of individual classes and mechanisms that developers can reuse or adapt [6]. It is characterized by three important features [23]:

First, a framework exhibits inversion of control (IoC) at runtime via callbacks to component hook methods after the occurrence of an event such as a mouse click or data arriving on a network connection. When an event occurs, the framework calls back to a virtual hook method in a pre-registered component which then performs application-defined processing in response to the event. The virtual hook method in the components decouples the application software from the reusable framework software, which makes it easier to extend and customize the applications as long as the interaction protocols and quality properties are not violated.

Second, a framework provides an integrated set of domain-specific structures and functionalities. Reuse of software depends largely on how well frameworks

---

4 http://www.zeroc.com
model the commonalities and variability in application domains. By leveraging the domain knowledge and prior efforts of experienced developers, frameworks embody common solutions to recurring application requirements and software design challenges that need not be recreated and re-validated for each new application.

Finally, a framework is a semi-complete application that programmers can customize to form complete applications by extending reusable components in the framework. In particular, frameworks help abstract the canonical control flow of applications in a domain into architectures and families of related components. At runtime, these components can collaborate to integrate customizable application-independent reusable code with customized application-defined code.

2.3. Design patterns

The design of a groupware framework can be greatly improved by using design patterns. A design pattern is a description of communicating objects and classes which is customized to solve a general design problem in a specific context. Each pattern represents a common and recurring design solution which can be applied over and over again in different problem-specific contexts [11, 35].

Patterns provide the designer with: a) abstract templates on how to make specific parts of a framework more flexible towards changes, b) a mechanism to document the architecture of a framework using a high abstraction level vocabulary, and c) a mechanism to impose rules about how to reuse or extend the framework, i.e., outline a specific interface on how to incorporate extensions. On the other hand, they provide the developer with a) a higher level of documentation for a complex framework consisting of numerous heavily interconnected classes and objects, and b) a guidance on how to extend the framework with new variations and whether or not extensions can be made [38].

2.4. Model View Controller Architecture

One of many important architectural pattern concepts that assists in the development of applications that have to display an interface to a user of a system is the Model View Controller architecture (or MVC). MVC structure offers significant benefits for distributed applications, and being familiar with it will enhance your understanding of a far wider range of concepts, some of which use or have borrowed ideas from MVC.

MVC is a design concept that attempts to separate an application into three distinct parts. One part is concerned with the actual work the application performs, another part is concerned with displaying the data or information of an application, and another part coordinates the former two parts in order to display the correct interface or execute some work that the application needs to complete. These parts are respectively:

- Model: The Model represents the part of an application that actually does the work and models the real-world problem that the application attempts to solve.
• View: The View is a representation of the Model that the user can see and interact with.

• Controller: The Controller is the part of the application that responds to commands from the user, deciding how the Model should be changed or retrieved and which View should be displayed.

The logical separation of the application into these parts ensures that the Model layer knows nothing about how it is displayed; it is restricted to just representing the component parts of the problem being solved by the application. Likewise, the View layer is concerned only with displaying data and not with implementing business logic, which is handled by the Model layer. The Controller, much like a traffic officer, directs the views to be displayed and fires off data changes and retrievals from the Model layer.

3. CIR in perspectives

CIR research could be subdivided in many ways, although it seems that there are five main areas of research that make up a considerable portion of the subject. They are: division of labor, sharing of knowledge, group awareness, session persistence, and evaluation. In the next items we shall briefly describe each one.

• Division of labor - Morris’s survey in [27] describes ad-hoc methods to avoid duplication of effort during a searching task, such as distributing the space of potential keywords, searching engines or sub-tasks among different group members. The supporting mechanisms for distributing and sharing work among participants are important for the success of a UI for multi-user search.

• Sharing of knowledge - In any collaborative setting, there will be a large and diverse knowledge base shared among groups of members. Each one will bring their own experience, expertise and topic knowledge to a particular searching task. What is needed is a way to enable the sharing of knowledge within the group [9].

• Group awareness - Awareness is an essential element in distributed collaborative environments. Over the last decade, a number of researchers have explored the role of group awareness for supporting collaboration between distributed groups. Specifically in CIR, awareness is another key requirement [27].

• Session Persistence - Storing a search session in a persistent format is a key requirement for facilitating collaboration during the session, revising the search at a later time, or sharing the results of a search with others [27].
Evaluation - Golovchinsky et al. remark in [14] that the evaluation is another challenge in CIR because traditional IR systems are usually evaluated based on recall and precision measures, and these metrics assume a single logical searcher (even if more than one person contributed to the final search results). Naïve approaches such as averaging by the number of participants do not reflect the reality that marginal rates of finding documents often decrease with the addition of more searchers. Furthermore, different team members may contribute in different ways, depending on their roles. In [8] Foley et al. present a measure to capture the quality and diversity across collaborating users’ ranked lists. These authors propose to measure the total number of unique relevant documents across users’ ranked lists at a certain cutoff and use this figure as a group score.

Recently, some researchers have identified different search scenarios where it is necessary to extend the IR systems with collaborative capabilities. For example, in the Web context, SearchTogether [28] is a system which enables remote users to synchronously or asynchronously collaborate when searching the Web. It supports collaboration with several mechanisms of group awareness, division of labor and persistence. CoSearch [1] is a system to improve the experience of co-located collaborative Web search by leveraging readily available devices such as mobile phones and extra peripherals. In the interactive multimedia context, Fischnlár-DiamondTouch [37] uses a conventional back-end search engine for text and image searching and a novel multi-user collaborative tabletop interface to support collaborative group searching. MediaMagic [31] is an interactive video search system. It allows the searcher to quickly assess query results and easily pivot those results off to form new queries. Using MediaMagic, two or more users with common information needs search together, simultaneously. The collaborative system provides tools, user interfaces and, most importantly, algorithmically-mediated retrieval to focus, enhance and augment the teams search and communication activities. In the educational context, Daffodil [20] is a virtual digital library system targeted at strategic support of users during the information search and retrieval process, which provides basic collaborative

<table>
<thead>
<tr>
<th>Application domain</th>
<th>Research example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaborative Web search</td>
<td>SearchTogether[28], S3[26], CIRE[33], Annotate[12], CSCW3[18], GroupWeb[16]</td>
</tr>
<tr>
<td>Co-located collaborative Web search</td>
<td>CoSearch[1]</td>
</tr>
<tr>
<td>Personalization in CIR systems</td>
<td>PERCIRS[30], METIORE[3]</td>
</tr>
<tr>
<td>Collaborative knowledge production in learning environments</td>
<td>Daffodil[20], K3[17]</td>
</tr>
<tr>
<td>Co-located collaborative multimedia search</td>
<td>Cerchiamo-MediaMagic[13, 31], Fischnlár-DiamondTouch[37]</td>
</tr>
</tbody>
</table>

Table 1: Examples of CIR Prototypes and Systems.
services and recommendations. In the table 1, we present a list of CIR examples that have been proposed in the last few years, pointing out their application domain.

4. Design considerations of CIRLab

This section describes the design of CIRLab. It is composed of three main modules (Figure 2):

- The Core. It supports various modern state-of-the-art collaborative IR techniques that can be reused or extended in a distributed collaborative environment;
- The Facades Mediator. It allows an easy integration between the core and front-end application; and
- The Actions Tracker which allows researchers to design and put into practice experiments on the basis of all that occurred during the collaborative search sessions.

With CIRLab it is easy to build front-end applications using the CIRLab Client Core. It permits the remote use of some service deployed on the server, such as instant messaging, sharing results, recommends relevant results, user’s query histories and other state-of-the-art CIR techniques. With Facades Mediator, it is possible to integrate the front-end client applications with the CIRLab Server Core by an efficient distributed notification mechanism, guaranteeing synchronous and asynchronous capabilities. It is also possible to build a front-end server application to manage the service deployed on the server, which can be used by researchers to configure parameters, index test collections and design
experiments or simply to manage a real CIR system. To complement both, it makes persistent important information for users’ help and user’s behavior for researchers. CIRLab includes the Actions Tracker module which implements some persistence mechanisms.

4.1. CIRLab Core

In this section we present the CIRLab basics. Note than when we say CIRLab core we refer to the basic elements of both - server and client. In order to differentiate between these two part, when needed, we shall use the terms Server Core and Client Core, respectively.

4.1.1. Distributed platform

The distribution and communication facilities of CIRLab are ICE conforming. Its middleware platform provides tools, APIs, and library support for object-oriented client/server applications development. In addition, we consider that the MVC pattern lends itself particularly well to the CIRLab’s distributed platform architecture, which, like traditional GUI applications, is almost entirely user-driven and has to present some sort of interface for that to occur.

In Figure 3, we present the MVC pattern applied in CIRLab’s architecture. The flow of events can be as follows. First, a group member can request the Collaborative Search Session Controller (MVC Controller) at the server side,
through the Facades Mediator (see Section 4.2), as well as some indication of
the work required wrapped in different parameters of the Request object. For
example, \texttt{action=divisionOfLabor, principle=metasearchAndSplit}, where \texttt{action}
represents the work that the controller should undertake and \texttt{principle} how it
must be done. Second, the controller receives the request and determines the
work to be completed based on the request. The Controller executes calls on
the CIR Model to undertake the required logic. Third, the Model component is
instructed to provide a list of objects needed by the Controller. It may access
some sort of collections or a persistence layer such as a database to provide
this. Fourth, the Controller is provided with the parameters wrapped in the
Response object to be displayed in the Collaborative Virtual Environment View
(MVC View) through the Facades Mediator. The Controller also determines the
appropriate View to provide for the end-user. Using the request dispatcher, the
Controller can provide the list of objects to the View selected. Fifth, the View
now has a reference to the data provided and renders the display of the list in
accordance with its display logic. Finally, the results of this process is provided
back to the group members in the response.

4.1.2. Collaborative Search Session Manager

A collaborative search session (CSS) is a group of end-users working together
to satisfy their shared information needs. Group manager is a common problem
that has been explained in many groupware. In CIRLab we adapted the proposal
by Gonzaga et al. in [15]. CIRLab core provides the appropriate mechanism for
managing the group members and their roles in a CSS according to the following
classification of users: a) Potential Member, whose elements are equivalent to
those from the end-user of a traditional IR system (e.g. Google, Yahoo! or
Windows Live Search in the World Wide Web context) and may manipulate
the instant messaging with other potential members through the Collaborative
Portal; b) Member is a subclass of Potential Member, whose elements have right
access to the CIR techniques and may interact with other members inside their
CSS; and c) Chairman is a subclass of Member, whose elements coordinate the
search session (each session has one chairman at the same time).

Because of potential conflicts that may arise from simultaneous access in a
CSS by multiple end-users, concurrency control to avoid/solve resources con-
tention is required.

For each collaborative search session, CIRLab allow the chairman to estab-
lish:

- **Integrity criteria.** The validity condition for minimum and maximum num-
  ber of members in a CSS (e.g. one CSS requires at least two members to
  be established; otherwise it is released or suspended). It can be: a) hard
  – the CSS is released if the integrity criteria are not satisfied; and b) soft
  – the CSS is suspended if the integrity criteria are not satisfied.

- **Membership Policy.** It establishes how a potential member joins and leaves
  a session. All potential members can negotiate an invitation to join a
session through the Collaborative Portal in different ways: a) static – a potential member must join a CSS by previous negotiation and before the work has started; b) dynamic and closed – each potential member must by explicitly invited to join the CSS; and c) dynamic and open – potential members can join a CSS by invitation or by own initiative at any time.

- Division of labor principle. It determines which principle to use in order to divide the search results among group members: a) Meta-search engines and split – the results of each search engine available are merged into one list which is automatically distributed among group members; b) Multi-search engines and switch – the results of each search engine available are switched among group members where one member can review only the results of a specific search engine; and c) Single-search engine and split – the results of the selected search engine are automatically distributed among group members;

4.1.3. IR Engines Manager

Another important component in CIRLab Server Core is the IR Engine Manager in charge of communicating and managing all search engines available to index and search the collections. It will distribute the query to each search engine, obtaining the results. This piece of software allows developers to use different search engines, such as Apache Lucene, Lemur/Indri and Terrier, for example, or any other designed specifically, in a transparent manner. In addition, it is possible to extend the number of IR engines available using the template method design pattern.

4.2. Facades Mediator

Facades Mediator is an event-driven notification service that has been developed in order to complement the CIRLab MVC-based architecture. MVC approach is largely based on an event-driven environment in which the end-user drives the flow of the application by using the interface. It is not surprising, therefore, to find its roots in the earliest graphical user interface (GUI) developments. Facades Mediator permits an agile development of the interactive collaborative virtual environment in a particular context. It supports notification mechanisms for instant messaging, different variants of division of labor, group awareness techniques, explicit recommendations and users’ manager. Each collaborative activity related to GUIs needs to use the Facades Mediator API. In addition, it is possible to extends the Facades Mediator with synchronous or asynchronous collaborative tasks using the template method design pattern.

4.3. Action Tracker

This module allows to make persistent all data that result significant for CIR techniques and researchers experimentation. User profiles can be used and may include demographic information, e.g. name, age, country, education level, etc., and may also represent the interests or preferences of either a group of users or a single person. In order to construct an individual user profile, information may
be collected explicitly through direct user intervention, or implicitly, through agents that monitor user activity.

5. An approach to CIRLab instantiation: Collaborative search-driven development

In this section we shall illustrate the CIRLab instantiation through a case study of CSDD with the purpose of describing how this framework can be used. CIR systems can be applied in several domains, such as travel planning, organizing social events, working on a homework assignment, medical environments or multimedia contexts. With the aim of beginning the pragmatic CIRLab validation, we identified software development as a possible applicability field where much evidence of programmers collaboration on a development process can be found. For example, concurrent edition of models and processes requires synchronous collaboration between architects and developers who cannot be physically present at a common location [22].

In the last few years, some IR systems have become a critical tools for software developers. Currently, there are some specialized IR systems for source-code search. Examples include Google Code Search, Krugle, CodeFetch, Koders, and Codase. Some of them, such as Koders, can be integrated with IDEs such as Eclipse and Visual Studio.NET, but all those interfaces are one-person oriented. More recently, there have been some significant efforts both from academia and the industry to fix Search-Driven Development (SDD) as a new research area. K. Krugler and J. D. Mitchell remark in [24] that “about 25% of a developer’s time is spent searching for information. It’s well spent, though – finding reusable code can get a project done on time and with high-quality results”.

The origin of this case study was the result of a brief survey that we offered to 50 students and professors engaged in work on software development projects in higher education in the domain of information technologies. Our survey results indicate that the collaborative SDD is a common task, but, unfortunately, there are no tools that adequately support it. Other surveys have been used to substantiate with evidence that the collaboration is a common activity in different search scenarios such as [27].

In the survey referenced above, when we asked: Have you ever collaborated with other programmers to search source-code?, 78.0% responded yes. In addition, we asked: Which are the activities that have motivated you to collaborate during the source-code search?; the most common answers were: a) meetings of the team members to clarify programming doubts while someone searches for source-code examples, b) dividing the search by each team member and sharing the final results, c) saving and documenting the search results of each one for sharing them, and d) consulting or answering doubts via chat or e-mail.

With this setting in mind, we have instantiated CIRLab to perform source-code searches. Figure 4 shows the prototype client interface. The search box (a) wraps the search control panel (SCP); it permits to specify the developers queries, programming language or project on which the search will be accomplished. Moreover, it can specify a searching field: comments, source-code, class
or methods declaration, and whole source files. Rather interestingly, source-code search all by itself doesn’t solve the whole problem. We need all of the technical information around and about the source-code to be able to really fly. For instance, the best examples of how to use some piece of source-code is often embedded as a small source-code snippet inside a magazine article, or even in the official technical documentation [24]. For these reasons, in the SCP you can select documents or the Web as collections, too. The SCP also offers the possibility to specify the division of labor principle. It determines which principle to use to divide the search results among team developers. The options box (b) wraps the principal options of the prototype that permits dynamic management of the GUI. For example, a developer can show: i) a collaborative portal where the developers can negotiate the creation of a collaborative search session and then show the embedded instant messaging box (c), ii) a recommendations panel box (d) to carry out explicit recommendations among developers, and iii) to add and show comments on the current and historical search results. In the search result box (e) the individual results and the recommendations made by others developers are shown. The previewer panel box (f) allow to review the results. The information box (g) permits to obtain specific information from the document selected in the results panel and determine its relevance.
6. Experimental evaluation

One of the objectives of this paper is to compare different search scenarios \( (SS_i) \) on SDD (see Table 2) using a common criteria to evaluate them. Each scenario could be obtained by considering a different division of labor techniques, search engines results, traditional tools that support collaboration, such as chat and email, and user views for IR and CIR systems. The different hypotheses around each scenario will be formulated by viewing these components isolately or even mixed. Our main hypothesis is that the collaborative work should help to improve the retrieval performance. The aim is to find the scenario that produces better results. Previously to discuss the results, we shall describe the different search scenarios and also the framework of the experimentation and the evaluation methodology that we have put into practice.

**Search Scenarios.** In this experimentation we are going to use 6 different search scenarios based on CIRLab (see Table 2) depending on the client tools used and the search principles considered. Thus, focussing on the client tools we can distinguish between Distributed View for Source-code Search (DV4SS) and the Individual View for Source-code Search (IV4SS). In the case of the individual view, the members of the groups collaborate using instant messages and/or email (Simple Messenger for Explicit Collaboration (SM4EC) and Simple Email for Explicit Collaboration (SE4EC), respectively).

In this experimentation we have considered two different search engines: Terrier and Apache Lucene. Three different search strategies, Simple Search (SSE), MEta-SEarch (MESE) and MUlti-SEarch (MUSE), has been implemented upon these search engines. Two different alternatives have been considered to distribute the search results (in a distributed environment): SPlit Algorithms (SPA) and SWitch Algorithms (SWA). In the first case, the results are automatically distributed among group members and in the the second a given member can review only the results of a specific search engine.

In order to configure these scenarios we have installed one server application and multiple clients applications. The clients applications are configured with a specific search scenario.

<table>
<thead>
<tr>
<th>Search scenario</th>
<th>Client tools(^{*})</th>
<th>Search principle(^{(#)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>( SS_1 )</td>
<td>( a )</td>
<td>1, 2</td>
</tr>
<tr>
<td>( SS_2 )</td>
<td>( b, c, d )</td>
<td>1</td>
</tr>
<tr>
<td>( SS_3 )</td>
<td>( a )</td>
<td>3, 4</td>
</tr>
<tr>
<td>( SS_4 )</td>
<td>( b, c, d )</td>
<td>3</td>
</tr>
<tr>
<td>( SS_5 )</td>
<td>( a )</td>
<td>5, 6</td>
</tr>
<tr>
<td>( SS_6 )</td>
<td>( b, c, d )</td>
<td>5</td>
</tr>
</tbody>
</table>

\(^{*}\) a:DV4SS,b:IV4SS,c:SM4EC,d:SE4EC  
\(^{(#)}\) 1:SSE,2:SSE&SPA,3:MESE,4:MESE&SPA,5:MUSE,6:MUSE&SWA

Table 2: Experimentation scenarios.
Framework. We shall consider each search scenario when a common task is proposed to a group of users: Solve the following problem of basic geometry using the Java programming language.

**Problem:** Calculate the perimeter of each of the polygons of a set of convex polygons. The input data are the coordinates of the vertices and are found in a text file. The results should be stored in another text file.

The users are students belonging to a technical institute of information science which are familiar with the C++ programming language, without Java background. They have to find classes and methods needed to solve the proposed problem with Java. Particularly, as target groups, we formed 6 teams of 3 students each one. Each group could use only clients applications with the same configuration. For example, the group 1 use clients applications configured with the search scenario SS1, group 2 use clients applications configured with the search scenario SS2 and so on.

The performance of each search scenario will be analyzed in order to evaluate our proposal. As mentioned in Section 4, the evaluation is a current challenge in CIR. In order to measure the effectiveness of the described prototype we shall consider as a base the metrics proposed by Pickens et al. in [32], i.e. viewed precision ($P_v$, the fraction of documents - seen by the user - that were relevant) and selected precision ($P_s$, the fraction of documents - judged relevant by the user - that were marked relevant in the ground truth), and selected recall/viewed recall ($R_s/R_v$) as their dependent measures. We extend this metric with the effort of interaction (EoI):

\[
EoI = \sum_{i=1}^{U} \frac{E_i}{U \times T},
\]

i.e. the fraction of explicit interaction of each user ($E_i$, as such explicit recommendations count and instant messages count) and users count ($U$) multiplied by the CSS time duration ($T$). Then, a good CIR system should have high $P_v$, $P_s$, $R_v$, $R_s$, and low $EoI$.

Data set and Relevance Judgements. In this framework, we have used as test collection a portion of the Java API, specifically the packages `java.lang`, `java.io`, `java.util` and `java.math`. We have focused on these packages because they are directly related to the context of the experiment.

In order to compute the recall and precision measures, and therefore, to measure the retrieval effectiveness, we have to assess the relevance judgements over the set of documents in the collection. This process have been done in the following way:

1. Generate a predefined set of queries strongly related to the objective of the experimentation: We use two techniques, first we use a group of 40 students with similar background to the target groups (technical institute of information science familiar with the C++ programming language) to
propose them the problem. We have to note that this group will not participate in the real experimentation, it just participated in this step to help us to identify appropriate queries.

This group used Google and made a list of used queries. Some of them were the following: "characters sequence", "access array elements", "string", "numeric functions", "array", "close file", "array index", "numeric operators", "output file", "string buffer", "string parser", "list", "file input operations", "string characters", "sqrt", "mathematics operators", etc.

Then we extended this set of queries in a broader set of query pairs completing where is necessary. For example "input text file" with "output text file", and so on.

2. Determine a set of relevance judgments for each predefined query: From the set of queries predefined in the step 1, a group of specialists in the Java programming language generated relevance judgments for each query, specifying the relevant source-code files. A relevance judgment may be given to the name of a method, a class, a term that appears within the source-code file or queries related to the content of the collection. For example, given the query "string characters", it has been consider relevant the files `java.lang.String`, `java.lang.StringBuffer`, `java.lang.StringBuilder`, `java.util.StringTokenizer`, `java.io.StringReader`, `java.io.StringWriter`.

Evaluation. Once the target geometry exercise have been explained to the students, each group had to solve it (in terms of Java algorithms). The maximum time given for searching is 30 minutes. All the search logs (used queries and the explicit interactions between users) are processed automatically by the evaluation module embedded in the prototype server management. In order to compare the performance between the different groups we have selected the 15 different queries that were used by all groups, i.e.

\[
\begin{align*}
q_1 &= \text{"string"} & q_2 &= \text{"array"} & q_3 &= \text{"array index"} \\
q_4 &= \text{"list"} & q_5 &= \text{"string characters"} & q_6 &= \text{"root"} \\
q_7 &= \text{"collection"} & q_8 &= \text{"file"} & q_9 &= \text{"power"} \\
q_{10} &= \text{"square root"} & q_{11} &= \text{"input"} & q_{12} &= \text{"input file output"} \\
q_{13} &= \text{"text file"} & q_{14} &= \text{"mathematics power"} & q_{15} &= \text{"text file"}
\end{align*}
\]

With respect to the used metrics, in the paper we only present the values of $P_s$, $R_s$ (Table 3) and $EoI$ (Figure 5) since they can easily allow to appreciate the advantages of search scenario based on techniques CIR versus the search scenarios based on IR techniques. It was surprising that the values of view recall and precision $P_v$ and $R_v$ do not reveal significant differences between the different search scenarios so we have decided not to include them in this paper.

Table 3 shows for each query $q_i$ its performance in terms of selected precision and selected recall for each scenario. The rows entitled with $AVG_P$ and $AVG_R$ contain the average selected precision and average selected recall for each scenario. Last row in Table 3 presents the global Effort of Interaction (in the deadline time).

The results of our experiments are quite conclusive: The best results have been obtained with $SS_3$ scenario, i.e. using a distributed view for source-code
search with MEta-SEarch plus a SPlit Algorithm as search principle. We obtain the best mean values of recall and precision with less interaction effort. Also, if we analyze the distributed versus the individual view of search ($SS_1$ vs $SS_2$, $SS_3$ vs $SS_4$ and $SS_5$ vs $SS_6$) we can conclude that a considerable improvements (considering all the metrics) have been obtained using the distributed interaction mechanism. From our point of view these results represent a strong support for the use of collaborative strategies in the field of code search.

On the other hand, focusing on the individual view for code search ($SS_2$, $SS_4$ and $SS_6$), it seems to be better the use of a simple search engine ($SS_2$). We believe that by using a simple search engine the user might fit the proposed queries to the retrieval strategies. On the contrary, when using a distributed view the use of different search engines seems to be preferable (both $SS_5$ and $SS_3$ get better performance than $SS_1$).

Focusing on the EoI (see Figure 5) it seems that there is a different performance between individual and distributed views of searching. Thus, individuals models seems to need less interaction in the beginning, but greater efforts are needed in the subsequent stages. On the other hand, distributed models require greater coordination efforts at the beginning, which are compensated at later stages by obtaining more relevant information.

Finally, and as a conclusion, by this experimentation we show how CIR techniques are useful in the field of SDD, as they help to improve the retrieval effectiveness.
<table>
<thead>
<tr>
<th>QRY</th>
<th>SS₁</th>
<th>SS₂</th>
<th>SS₃</th>
<th>SS₄</th>
<th>SS₅</th>
<th>SS₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₁</td>
<td>0.0120</td>
<td>0.0080</td>
<td>0.0160</td>
<td>0.0040</td>
<td>0.0400</td>
<td>0.0200</td>
</tr>
<tr>
<td>q₂</td>
<td>0.0526</td>
<td>0.0385</td>
<td>0.0562</td>
<td>0.0461</td>
<td>0.0329</td>
<td>0.0263</td>
</tr>
<tr>
<td>q₃</td>
<td>0.1045</td>
<td>0.0597</td>
<td>0.0896</td>
<td>0.0448</td>
<td>0.1343</td>
<td>0.0299</td>
</tr>
<tr>
<td>q₄</td>
<td>0.0658</td>
<td>0.0395</td>
<td>0.0789</td>
<td>0.0132</td>
<td>0.0526</td>
<td>0.0263</td>
</tr>
<tr>
<td>q₅</td>
<td>0.0476</td>
<td>0.0833</td>
<td>0.0952</td>
<td>0.0357</td>
<td>0.0714</td>
<td>0.0238</td>
</tr>
<tr>
<td>q₆</td>
<td>0.1667</td>
<td>0.0556</td>
<td>0.1111</td>
<td>0.1667</td>
<td>0.1667</td>
<td>0.0556</td>
</tr>
<tr>
<td>q₇</td>
<td>0.3382</td>
<td>0.1176</td>
<td>0.3088</td>
<td>0.0882</td>
<td>0.2941</td>
<td>0.2647</td>
</tr>
<tr>
<td>q₈</td>
<td>0.2368</td>
<td>0.1842</td>
<td>0.3289</td>
<td>0.0263</td>
<td>0.1711</td>
<td>0.0132</td>
</tr>
<tr>
<td>q₉</td>
<td>0.2500</td>
<td>0.1875</td>
<td>0.2500</td>
<td>0.3125</td>
<td>0.1250</td>
<td>0.0625</td>
</tr>
<tr>
<td>q₁₀</td>
<td>0.3750</td>
<td>0.2500</td>
<td>0.3750</td>
<td>0.1250</td>
<td>0.2500</td>
<td>0.1250</td>
</tr>
<tr>
<td>q₁₁</td>
<td>0.2667</td>
<td>0.2400</td>
<td>0.2933</td>
<td>0.0800</td>
<td>0.1600</td>
<td>0.0533</td>
</tr>
<tr>
<td>q₁₂</td>
<td>0.3200</td>
<td>0.2000</td>
<td>0.4000</td>
<td>0.1200</td>
<td>0.3200</td>
<td>0.0800</td>
</tr>
<tr>
<td>q₁₃</td>
<td>0.1250</td>
<td>0.2500</td>
<td>0.5000</td>
<td>0.1250</td>
<td>0.6250</td>
<td>0.3750</td>
</tr>
<tr>
<td>q₁₄</td>
<td>0.2857</td>
<td>0.4286</td>
<td>0.5714</td>
<td>0.2857</td>
<td>0.5714</td>
<td>0.4286</td>
</tr>
<tr>
<td>q₁₅</td>
<td>0.4444</td>
<td>0.3333</td>
<td>0.6667</td>
<td>0.2222</td>
<td>0.2778</td>
<td>0.0556</td>
</tr>
<tr>
<td>(\text{AV}\ E_P)</td>
<td>0.2061</td>
<td>0.1651</td>
<td>0.2763</td>
<td>0.1130</td>
<td>0.2195</td>
<td>0.1093</td>
</tr>
<tr>
<td>SELECTED PRECISION, (P_s)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QRY</th>
<th>SS₁</th>
<th>SS₂</th>
<th>SS₃</th>
<th>SS₄</th>
<th>SS₅</th>
<th>SS₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₁</td>
<td>0.1250</td>
<td>0.0833</td>
<td>0.1667</td>
<td>0.0417</td>
<td>0.4167</td>
<td>0.2083</td>
</tr>
<tr>
<td>q₂</td>
<td>0.3810</td>
<td>0.2857</td>
<td>0.4286</td>
<td>0.3333</td>
<td>0.3381</td>
<td>0.1905</td>
</tr>
<tr>
<td>q₃</td>
<td>0.5000</td>
<td>0.2857</td>
<td>0.4286</td>
<td>0.2143</td>
<td>0.6429</td>
<td>0.1429</td>
</tr>
<tr>
<td>q₄</td>
<td>0.8333</td>
<td>0.5000</td>
<td>1.0000</td>
<td>0.1667</td>
<td>0.6667</td>
<td>0.3333</td>
</tr>
<tr>
<td>q₅</td>
<td>0.5000</td>
<td>0.8750</td>
<td>1.0000</td>
<td>0.3750</td>
<td>0.7500</td>
<td>0.2500</td>
</tr>
<tr>
<td>q₆</td>
<td>0.3333</td>
<td>0.1111</td>
<td>0.2222</td>
<td>0.3333</td>
<td>0.3333</td>
<td>0.1111</td>
</tr>
<tr>
<td>q₇</td>
<td>0.6970</td>
<td>0.2424</td>
<td>0.6364</td>
<td>0.1818</td>
<td>0.6061</td>
<td>0.5455</td>
</tr>
<tr>
<td>q₈</td>
<td>0.6923</td>
<td>0.5385</td>
<td>0.9615</td>
<td>0.0769</td>
<td>0.5000</td>
<td>0.0385</td>
</tr>
<tr>
<td>q₉</td>
<td>0.6667</td>
<td>0.5000</td>
<td>1.0000</td>
<td>0.8333</td>
<td>0.3333</td>
<td>0.1667</td>
</tr>
<tr>
<td>q₁₀</td>
<td>1.0000</td>
<td>0.6667</td>
<td>0.6667</td>
<td>0.3333</td>
<td>0.6667</td>
<td>0.3333</td>
</tr>
<tr>
<td>q₁₁</td>
<td>0.7143</td>
<td>0.6429</td>
<td>0.7857</td>
<td>0.2143</td>
<td>0.4286</td>
<td>0.1429</td>
</tr>
<tr>
<td>q₁₂</td>
<td>0.6667</td>
<td>0.4167</td>
<td>0.8333</td>
<td>0.2500</td>
<td>0.6667</td>
<td>0.1667</td>
</tr>
<tr>
<td>q₁₃</td>
<td>0.2000</td>
<td>0.4000</td>
<td>0.8000</td>
<td>0.2000</td>
<td>1.0000</td>
<td>0.6000</td>
</tr>
<tr>
<td>q₁₄</td>
<td>0.5000</td>
<td>0.7500</td>
<td>1.0000</td>
<td>0.5000</td>
<td>1.0000</td>
<td>0.7500</td>
</tr>
<tr>
<td>q₁₅</td>
<td>0.5714</td>
<td>0.4286</td>
<td>0.8571</td>
<td>0.2857</td>
<td>0.3571</td>
<td>0.0714</td>
</tr>
<tr>
<td>(\text{AV}\ E_R)</td>
<td>0.5587</td>
<td>0.4484</td>
<td>0.7191</td>
<td>0.2893</td>
<td>0.5737</td>
<td>0.2701</td>
</tr>
<tr>
<td>SELECTED RECALL, (R_s)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| \(30M\) | 0.6778 | 0.7000 | 0.4066 | 0.6000 | 0.5889 | 0.6333 |
| EFFORT OF INTERACTION, \(EoI\) | | | | | | |

Table 3: Experimental results.
7. Conclusions and future work

The CIR community has been developing some groupware in the last few years, but many of these developments address only specific problems or do not adequately support collaborative interaction. Therefore, it is convenient to have an experimental platform that allows to test new algorithms which use inputs from multiple users and compare their searching results with a single user activity. In that sense, we have described CIRLab - a groupware framework for CIR research and experimentation. We have shown its main features and its modular architecture which is easy to extend and reuse in different search contexts. The principal beneficiaries of this work will be researchers in the CIR field who are considering the development of multi-user search applications.

There are many contexts where expanding IR horizons towards CIR would be an advantage. Some CIR systems and prototypes on Web search, medical environment and multimedia have been proposed. This paper proposes the software development as another study area for CIR systems to deploy. Software development is a collaborative process where teams of developers work together to design, solve problems and, particularly, share their source code search results. We have analyzed the development of a CSDD prototype using CIRLab components. In addition, and in order to exemplify the instantiation of the platform in the SDD field, a user study has been conducted under this context with the aim of testing the viability of CIR versus the classic IR, as well as the different elements implemented in CIRLab. The main conclusion is the use of collaborative features improves the effectiveness of the search process.

In this sense, we shall focus on transforming our CSDD prototype in a plug-in to add it at some open-source IDE as such Eclipse or NetBeans. We have also shown a proposal for the evaluation of this prototype and other CIR applications.

Our proposal contemplates some recent ideas on CIR research. Session persistence, division of labor, knowledge sharing and group awareness show up within CIR as important elements to be considered because of their notable benefits to group members with shared information needs.

Some of these CIR elements can be mixed up with personalization techniques in order to improve the group search interaction. At the moment, we are working to mix CIR algorithms with personalization techniques, such as Groupization, Smart Splitting and Group Hit-Highlighting, proposed by Morris et al in [29], in order to allow a better integration between the different stages of the CIR process. To deliver recommendations, we identify Taste\(^5\) class library. Taste is a flexible, fast collaborative filtering engine for Java. This engine takes users’ preferences for items (tastes) and returns estimated preferences for other items. Taste is designed for performance, scalability and flexibility. On the other hand, and taking into consideration our survey results, when 92.9% of our respondents use their workstation as an important dynamic collection of

\(^{5}\)http://lucene.apache.org/mahout/taste.html
relevant information, we shall add to CIRLab the capability of indexing local collections, which can by used by the teamwork. This new feature of CIRLab may enhance the efficiency of CIR systems with respect to specific problems.

References


URL http://portal.acm.org/citation.cfm?id=99332.99343


URL http://dx.doi.org/10.1002/bult.2006.1720320508


URL http://dx.doi.org/10.1145/1390334.1390389


