ANDES: An approach to embed search services on the Web browser

Gabriela Bosetti, Alex Tacuri, Ishaya Gambo, Sergio Firmenich, Gustavo Rossi, Marco Winckler, Alejandro Fernandez

▶ **To cite this version:**

Gabriela Bosetti, Alex Tacuri, Ishaya Gambo, Sergio Firmenich, Gustavo Rossi, et al.. ANDES: An approach to embed search services on the Web browser. Computer Standards and Interfaces, 2022, 82, pp.103633. 10.1016/j.csi.2022.103633 . hal-03811257

HAL Id: hal-03811257
https://inria.hal.science/hal-03811257

Submitted on 11 Oct 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ANDES: an approach to embed search services on the Web browser

Gabriela Bosetti[a,*], Alex Tacuri[b,c], Ishaya Gambo[d,e], Sergio Firmenich[c,f], Gustavo Rossi[c,f], Marco Winckler[g], Alejandro Fernandez[c,g]

[a] *VeryConnect. Pavilion 1 12 Minerva Way. Glasgow. Scotland. G3 8AU*
[b] *Escuela Superior Politécnica de Chimborazo. Panamericana Sur km 1 y 1/2 . Riobamba. Ecuador*
[c] *LIFIA, Facultad de Informática, UNLP. La Plata. Calle 50 y 120, S/N. Buenos Aires. Argentina.*
[d] *Institute of Computer Science. University of Tartu. 51009 Tartu. Estonia*
[e] *Department of Computer Science & Engineering. Obafemi Awolowo University. Nigeria*
[f] *CONICET. Godoy Cruz 2290. Ciudad Autónoma de Buenos Aires. Argentina*
[g] *I3S, SPARKS team, Université Côte d'Azur. Sophia - Antipolis. France*
[h] *CICPBA. Calle 526 entre 10 y 11. La Plata. Buenos Aires. Argentina*

Abstract

This article addresses search tasks, which are the most frequent type of tasks users do nowadays on the Web. We suggest that the customization of search services would improve the user's searching experience, and it also might make search tasks more effective and efficient. We analyze user tasks for performing searches and types of search services. Then, we distinguish between primary search tasks (related to the initial user goal) and ancillary search tasks (those that are performed to find additional information necessary to understand how to achieve a primary search task). We also explain different types of search services, including global search services (which has an extensive search scope, possible the whole Web) and in-site search services (which has a narrow scope of search focused on a specific Web site or data set). We also discuss features embedded into Web browsers to support search tasks and how Web browsers

*Corresponding author
Email addresses: gabriela@veryconnect.com (Gabriela Bosetti), atacuri@espoch.edu.ec (Alex Tacuri), ishaya.gambo@ut.ee (Ishaya Gambo), sergio.firmenich@lifia.info.unlp.edu.ar (Sergio Firmenich), gustavo@lifia.info.unlp.edu.ar (Gustavo Rossi), winckler@unice.fr (Marco Winckler), alejandro.fernandez@lifia.info.unlp.edu.ar (Alejandro Fernandez)

can support extensions allowing users to integrate customized search services. We propose an approach called ANDES that allows users to integrate into the Web browser new search services that are created by customizing search services offered by existing Web sites. By means of Web augmentation techniques, we demonstrate how users can customize the browser to trigger search services to any Web site and integrate the Web page results. We report on two evaluations showing that end-users can specify and use search services following our approach and that the proposed interaction is more convenient in contrast to traditional mechanisms. Beyond the tool used to demonstrate the feasibility of the ANDES approach, this paper aims to discuss new strategies for end-user programming that allow users to customize search services over the Web.
Keywords: Web search, end-user programming, user experience, Web augmentation

---

1. Introduction

Users might find information over the Web using either searching or browsing tasks. In searching tasks, users are looking for a specific result (goal-directed tasks), while in browsing tasks, users are looking for general ideas or multiple answers (exploratory tasks). In this paper, we focus on searching tasks as this is the most common way to find information over the Web nowadays. Indeed, the numbers describing search tasks over the Web are astonishing. According to Alexa Internet [1], google.com is the most visited Web site, where users spend an average daily time of 15 minutes to reach an average of 17.25 pages. Whilst search becomes a cornerstone of the Web experience, not all search services look alike.

In terms of scope, Google.com is a good example of what we call a "global search service", as it crawls and indexes most of the visible Web. Nonetheless, searching is not limited to global search services; a large number of the queries

[1] https://www.alexa.com/topsites Last visited on February 9th, 2021

made by users are handled by specialized, on-site search services offered by most Web sites. Because "on-site search" has a narrower scope, the results do not compete for the user's attention with results from all over the Web. A broad user study conducted by Yahoo and Google researchers found that 9% of all user activity corresponds to searching and, of that, about 60% represents global Web search, whereas the remaining 40% refers to on-site search [**?**]. Both global search and in-site search are valuable strategies for finding information, and circumstantial users' needs determine their best use.

The traditional way of searching the Web involves choosing a search service (by visiting a Web site), formulating a query (typing keywords on an input box), and navigating the results (often shown in the same window/tab where the query was formulated). Modern browsers offer embedded search mechanisms so that users can formulate queries directly in the so-called Omnibox (see Figure 1) without having to visit a Web site to access the search service. Users can configure the Omnibox so the queries can be directed to an on-site search service (see Figure 2). For that, when users navigate sites like Amazon.com in Firefox, they see a "plus" icon on the left side of the Omnibox, as shown in Figure 3; the sign indicates that users can add such search service with just a click. Nevertheless, the configuration search mechanisms on the browser is limited and not available for all Web sites. This article provides the means so end users can use any on-site search mechanisms by using Web augmentation technologies.

Understanding the user's needs and behavior while searching is essential to improve the tools that support search tasks [**?**] [**?**]. According to Broder [**?**] a Web search can either be navigational, transactional, or informational. The purpose of a navigational Web search is to find the URL of a known Web site; the purpose of a transactional one is to reach a Web site in which to conduct further interaction; and the purpose of an informational Web search is to find information assumed to be available on the Web. Informational Web searchers are similar to those in traditional information (document) retrieval. The use of the appropriate searching strategy might help users to save time to find information. One of the main problems of existing search services is that

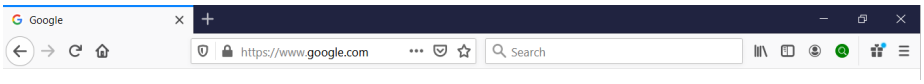

Figure 1: Filling in forms for triggering queries directly from the toolbar

.

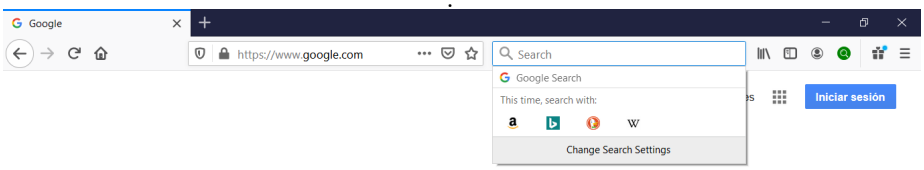

Figure 2: Configuration of search services from the toolbar

.

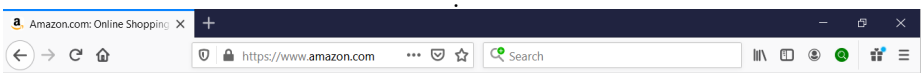

Figure 3: Specialized search mechanism according to the Web site visited by the user, ex. for Amazon

.

they are proposed as independent applications that do not take into account the context of use of the query results. Moreover, existing search services are quite inflexible and offer very little room for customizing the scope of the search and the format of the query results.

While some informational searches can be directly linked to a user's primary goal (e.g., find articles to learn about COVID vaccines), other informational searches are nested in other search tasks. These nested searches, also called "ancillary searches" [**?**], serve the purpose of finding information to complement what is being read without diverting the users' attention from their primary task. A simple example of an ancillary search is looking up a term's definition (ancillary tasks necessary to understand the meaning of a particular word) while reading an article (the primary user tasks). If the ancillary search is performed in the same browser window (replacing the content that motivated the search), the context of the primary task is lost. For that reason, users often keep multiple tabs or windows open in parallel to track their context [**?**] when performing multiple searches over the Web. Unfortunately, multiple tabs divide the users' attention since they have to remember their contents while navigating between

tabs. Some authors that tackle this problem [**?**] suggest that the results of the ancillary search should be displayed on the same page (e.g., as a floating panel).

In this paper we propose an approach that allows users to go a step forward into the creation of search service that can be customized to fulfill specific users' needs. Our approach called ANDES, describes how users can redefine the scope of search services, indicate the expected format of query results, and place the results in the context of use that best suit their needs. By using End-User Programming and Web augmentation techniques [**?**], a supporting tool called ANDES as well as the approach, allows users to integrate search services to trigger queries to these services without having to open a window/tab, and places the query results as a pop-up that floats over any Web (which is meant to reduce the articulatory distance between primary search tasks and ancillary search tasks). Moreover, search services created with ANDES can be shared with other users.

The present paper extends previous work [**?**] by providing a detailed analysis of user tasks and preliminary results of a user testing of the tools. We also analyze the trade-offs of having custom search services versus generic search services. Based on the preliminary results of user testing, we have drawn some conclusions and recommendations for the development of adaptable search services, and we suggest some research questions related to the user of end-user programming for developing search services over the Web.

The remaining of this article is organized as follows. Section 2 presents the concepts necessary to understand this work (Web Augmentation, and End-User Development). Section 3 presents an overview of the proposed approach, using an example and tasks models for primary and ancillary searches. Sections 4 and 5 provide additional details about the approach and its key design decisions. Section 6 presents the evaluation of the approach with users, featuring formative and summative evaluations. Section 7 discusses related works. Finally, section 8 presents conclusions and mentions further work.

2. Background

The tools proposed in this work build upon two basic concepts: Web Augmentation (WA) and End User Development (EUD). WA refers to technologies that modify Web pages on the browser to include features required by the users but not provided by the Web server. EUD refers to technology that allows users to create tools that fulfill their own needs.

2.1. Web Augmentation

Web Augmentation (WA) [**?**] refers to a set of tools and techniques to manipulate and enrich Web pages to meet the users' needs that the application developers did not originally consider. Augmentations may change the content, the structure, the style, or the behavior of an existing Web page, without it being necessary to modify the source code of the application that served it.

A Web extension allows to manipulate the Web document presented to the user. The manipulation can take place in the Web server [**?**], in an intermediary proxy [**?**], and the client browser [**?**]. The most frequent method is DOM manipulation via browser extensions and user scripts. Other manipulation approaches in the browser include bookmarklets [**?**] and native applications that enhance the DOM [**?**].

There are many repositories of user scripts and Web extensions available (ex. greasyfork.org) and more than one thousand extensions under the category "Search tools" in Google's Chrome Web Store; several of which are focused on retrieving results from specific Web sites. However, the use of WA script requires coding skills to a certain degree [**?**] [**?**].

The tool proposed along the approach ANDES is implemented as Web Augmentation scripts. The use of WA allows integrating search services created by the users directly into the browser. Moreover, it allows users to control where to place the query results in Web pages.

2.2. End-user development

When tools match the tasks they are designed for, they go unnoticed by the user. However, when applications fail to meet the user's requirements, it

becomes necessary to develop a new one or customize the one available.

Previous works [**?**] [**?**] have demonstrated that, if appropriate tools are provided, end users might be able to create what they need (or at least define more precisely part of what they need). To describe such as special type of software, Lieberman et al. [**?**] use the term End User Development as: "a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact".

It is worth mentioning that Web augmentation has been studied as a promising means for End-User Development [**?**]. EUD environments allow users with basic software construction skills to the author or customize applications. In comparison to professional software development environments, EUD environments work at a higher level of abstraction; they frequently rely on abstract syntax [**?**], in many cases even implementing an entire domain-specific language, as in the case of Sticklet [**?**]. Studies demonstrate that users are willing to have an active role in developing the Web, they do not always want to depend on a developer, and they want to create their solutions [**?**] [**?**].

The ANDES approach is a case of End-User Development as users must program somewhat the search services to determine the scope of the search and the format of the results. Nonetheless, the programming required by ANDES follows a visual program paradigm and does not require coding skills.

3. The ANDES approach: an overview

The ANDES approach is grounded on idea that search services can be perceived as ancillary search tasks that help users to perform a primary task. Hereafter we describe and compare these types of search tasks

3.1. Primary search tasks

Searching for a given Web page is a typical example of a primary search. For instance, to find The New York Times Web site, a user would use a search service like Google or Bing, with something like "the new your times" as the

Figure 4: Task model example for a primary search

search string. Then, if the user also wants to find the New York Public Library Web site, s/he must initiate another primary Web search, using the same set of interactions. The left-side of Figure 4 shows a task model using the HAMSTERS notation [**?**] to describe primary searches it in terms of user interactions. As we shall see, a search task is iterative and can be performed many times until the user find the results. A search is decomposed in sub-tasks "formulate a query", "process query" (by the system), "show results", (optionally) "refine results" (which might include "filtering"), and "stop search" which indicates the users found what he was looking for. It is interesting to notice that the model indicate that many search tasks can be run in parallel. The right-side of Figure 4 illustrates the scenario using screen shots.

3.2. Ancillary search tasks

An ancillary search task is a task that is triggered by the user to obtain information necessary to complete another primary task. Imagine you are a regular reader of The New York Times on-line. The newspaper has a section where authors discuss their literary preferences. You like this section and, once a week, you read it to find your "next book to read". Consider the article entitled "Allegra Goodman: by the book" as an example. Each time the article mentions a book that catches you eye, you wonder if it would be available at the New York Public Library (NYPL) Web site to borrow it, so you conduct an ancillary search. The problem with this ancillary search is that, for each book, you need to copy the name of the book, open a new tab, navigate to the NYPL

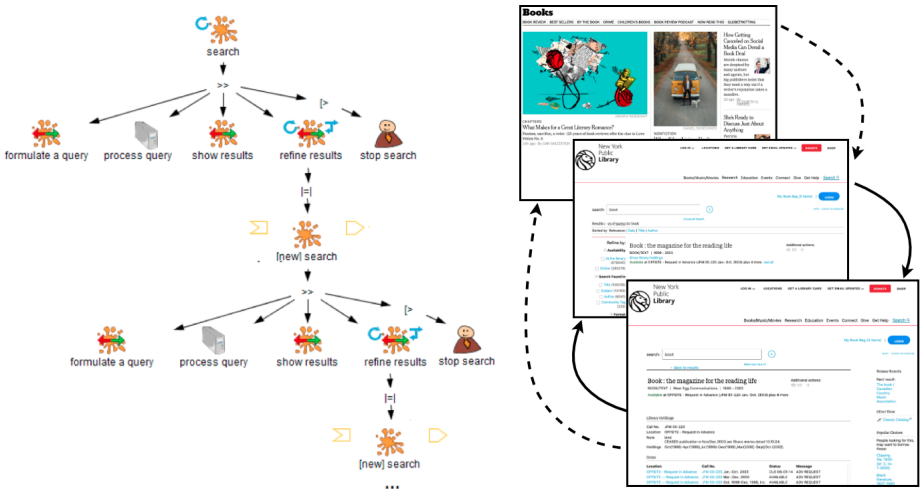

Figure 5: Task model example for an ancillary search

Web site, paste the title of the book into the input, and execute the search. That takes time, it is tedious and diverts your attention from the article you are reading. Figure 5 illustrates these tasks. The left-side of the figure shows a task model for such a scenario. As we shall see, the ancillary search task is a search task nested into a primary search tasks. The right-side of figure 5 illustrates a scenario for an ancillary task.

3.3. Example of primary and ancillary search tasks with ANDES

The ANDES approach put forward that it is possible to improve the users' experience while conducting ancillary searches by allowing users. 1) to create customized search services that, connected to specific Web sites, will provide the most relevant information to fulfill specific user's needs; 2) to trigger queries to those services from any Web pages without the need to open a new window/tab to run a search; 3) to integrate search results in the Web page that originates the search.

To illustrate this scenario, imagine that you could trigger your query to the NYPL search service by simply selecting the title of each book directly from the article of the New York Times Web site (shown in Figure 6). With a single click, you could see the results of the search as a table, floating over the article

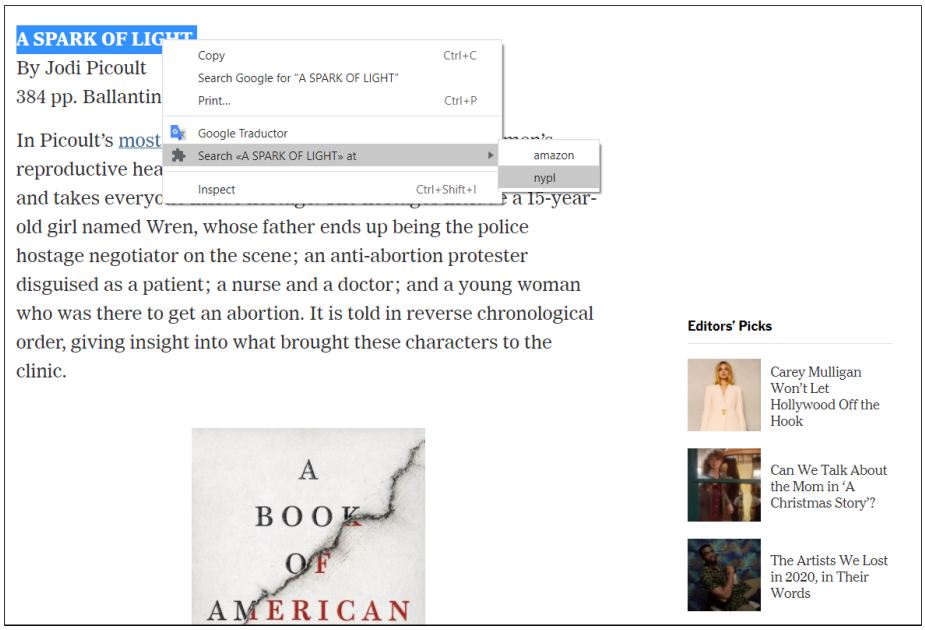

Figure 6: A contextual menu displaying custom search to the search service "NYPL"

you are reading (as shown Figure 7). You could sort the table by availability of the book, by the year of the edition, by the type of cover, etc. Wouldn't that be better? Moreover, what if instead of NYPL, you could customize your search to trigger queries to any search services you'd like (such as Netflix, YouTube, Vimeo, DBLP, Google Scholar, LinkedIn, etc.)?.

ANDES (ANcillary sEarch Services) is an EUD tool to improve ancillary searching. It is a Web add-on (aka a Web extension) that improves the way users deal with ancillary searches while navigating the Web, thus reducing the user's effort of dealing with the execution and the evaluation gulfs, as explained in [**?**]. With ANDES, the user can:

- Encapsulate an existing search service into a custom, object-based, ancillary search service. To achieve this, end-users rely on Web-annotations and point and click EUD strategies.
- Trigger an ancillary search from any page in the Web, using any of the already defined ancillary search services. To achieve this, end-users rely on

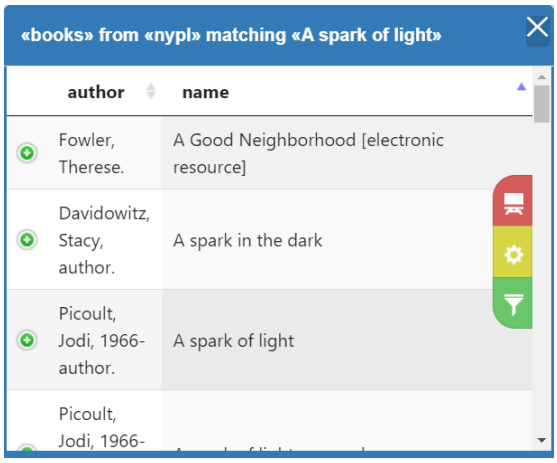

Figure 7: A floating panel presenting the results of an ancillary search

Web-annotations and point and click EUD strategies. Note that although in the context of EUD for automating user interaction (as a search) is very common to follow a record&play approach, we preferred to use an annotation&play approach because the ANDES search service may be defined at different sessions, under an evolutive approach in which the user may add behavior to it after a first version is defined, for instance, for adding pagination, filters, etc.

- Explore the results of the search without leaving the Web-page that originated it
- Share the definition of an ancillary search service with other users.

When the user performs a search using ANDES, he perceives a Web augmentation effect based on a floating table that is rendered in the current Web page, which shows the search results obtained from the desired Web site, as in Figure 8. To do that, ANDES automates the interaction required to navigate the target Web site, perform a search, and extract the results based on a domain object description.

Summarizing, there are two possible processes for accomplishing ancillary searches nowadays: using the browser's Omnibox (Figure 9-A), or opening a new

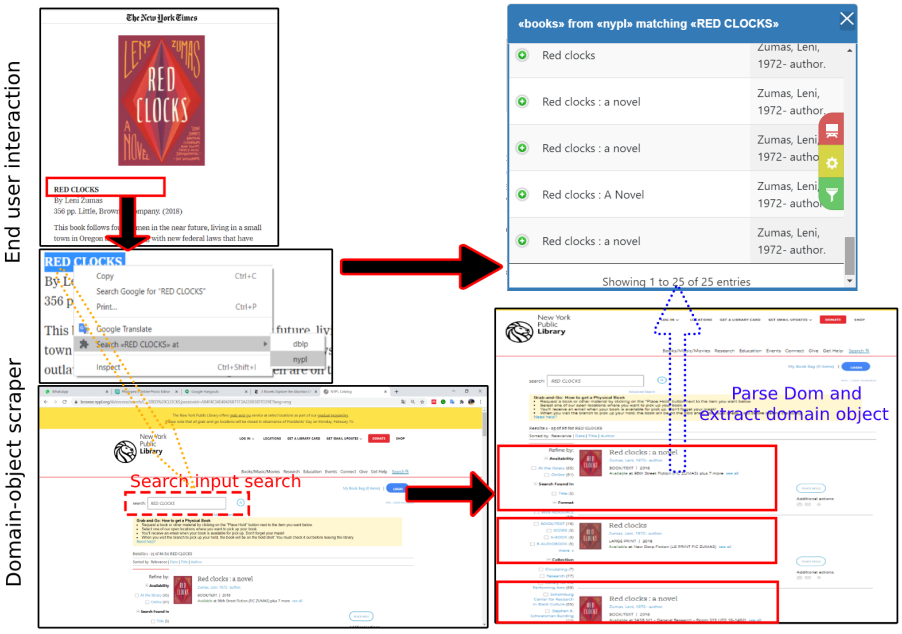

Figure 8: Ancillary search automation and results integration

tab for triggering any Web-based search engine available on the Web (Figure 9-B). We are proposing a new mechanism, which requires a one-time definition –via DOM annotations– of a Search Service (Figure 9-D), and then using the service –as many times as the user wants– in a similar way as an Omnibox search service, which is highlighting keywords in the DOM, triggering the search via contextual menu and having the ancillary SERP results in the same context as the main user task (Figure 9-C).

4. Using ANDES

The examples described below assume that the user has already installed the ANDES add-on in the browser; its source code is publicly available on GitHub [2] and there is a playlist with demo videos [3].

[2] The ANDES webextension: `https://github.com/gbosetti/web-ancillary-search`
[3] `https://www.youtube.com/playlist?list=PLHuNJBFXxaLA1FfFMtzvOXojIOyg4WWxj`

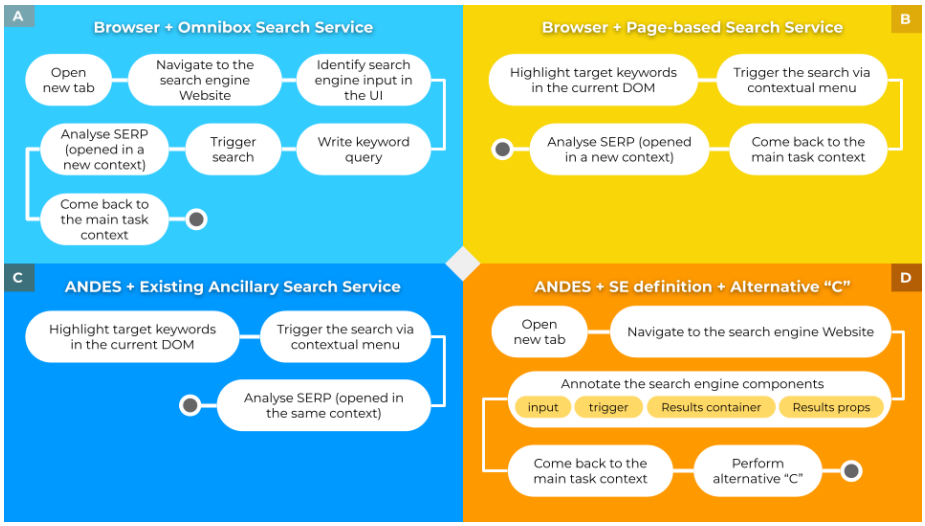

Figure 9: Existing and proposed mechanisms for ancillary searching

4.1. Defining an ancillary search service

To interactively create an ancillary search service, the user first navigates to a Search Service Results Page (SERP). Once in the SERP, the user clicks the ANDES toolbar button as shown in Figure 10 to start. A sidebar opens, floating over the SERP, which helps the user identify the search service components.

A welcome screen is presented to the user, as seen on step 2 of Figure 11, which includes the list of ancillary search services already defined for the site, if any. In the figure, the user is creating a first service for the NYPL; to do so, she clicks on the "New Service" button, and she is asked to provide a name for the new service (step 3). A name is suggested in the placeholder. The user clicks "Next" to continue.

Now the user must indicate which search form element is the input field for the search string. She clicks on the desired input field, which is highlighted to provide feedback (step 4). During this step, all search form elements are temporarily disabled (for example, clicking on the search button does not activate the search). The user clicks "Next" to continue.

Next, the user must indicate a strategy for triggering the search query (step 5). If the user chooses "click on a control", she must proceed to indicate a

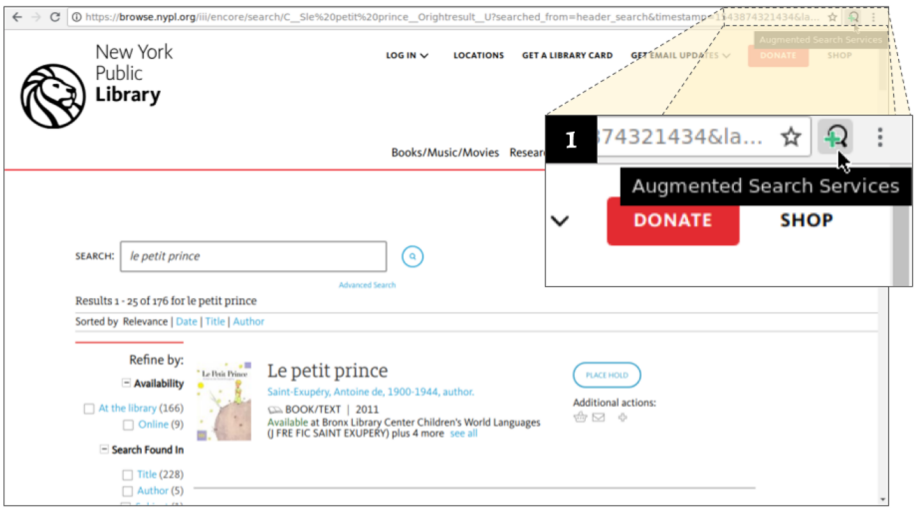

Figure 10: Accessing the tool from a SERP

control element (e.g., a button, a link) as she did to select the input field. Other triggering strategies are available, for instance, "press enter" (step 6) or "type and wait" (for those sites that automatically trigger a query while the user is typing). The user clicks "Next" to continue.

Finally, the user must indicate which elements in the SERP represent the results by pointing and clicking (step 8). Clicking on a DOM element marks the element and its siblings as results. ANDES is object-based, which means that searches return objects with types and properties instead of documents. Being object-based allows for richer manipulation and visualization of search results. However, it involves some additional steps during the definition of the ancillary search service. After results have been marked, the user must provide a type (which is Book, in step 8 of the figure). Then, by clicking in each of the DOM sub-elements of a result, the user defines the result's properties. In Figure 12, the user has selected the title and the author as the properties to extract from the results. Finally, she saves the created service (step 10).

Figure 11: Service definition: the input, trigger and results container selection

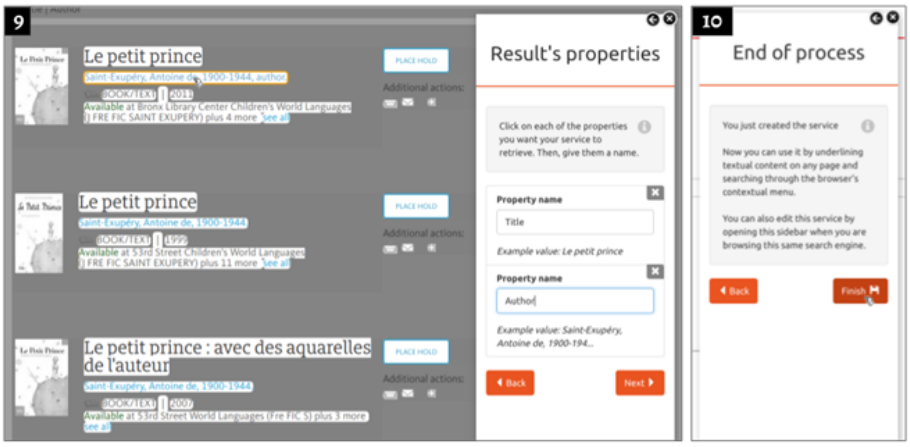

Figure 12: Service definition: the results' properties selection

4.2. Performing an ancillary search

Now that the ancillary search service was defined as explained in section 5.1, triggering the search requires only a few interactions. The user highlights a text she wants to search, opens the browser's context-menu (usually by right-clicking the text selection), and selects one of the available Augmented Search Services. Figure 13 depicts an example where the user is checking the availability in the NYPL Web site of two books listed in a GoodReads' ranking [4]. In the first step, she is using the "NYPL" service for searching a book matching the "A spark of light" keywords. In the second step, she is presented with a draggable pop-up element containing the retrieved results. Then, she repeats the operation with another book called "Save the Loonies" but, as shown in the third step, there are no results for this query.

5. The design of ANDES

ANDES (AugmeNteD sEarch Services) is a Web browser extension that allows end-users to model the search services offered by any Web sites so they can

[4] https://www.goodreads.com/group/bookshelf/52937-around-the-world-in-80-books

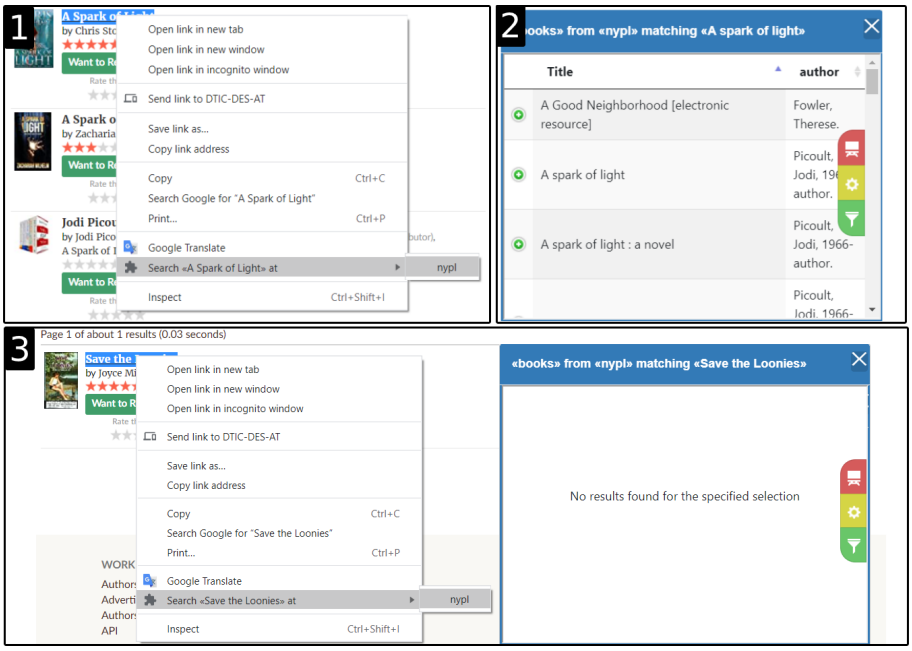

Figure 13: Using the NYPL Augmented Search Service

be used to support ancillary searches in other contexts. It consists of organized classes in three packages, as shown in Figure 14.

The "Metamodel" package groups classes that model search services. Every time a new Search Service model is created, its specification (constructed in terms of instances of the metamodel classes) is stored at the client-side.

The "Search Context" package groups the classes that represent a search query and its results. When the user wants to perform a search task, a SearchQuery is created (there are two strategies for doing that: text selection and text input). When the search results are retrieved, these are materialized as instances of the SearchResult class.

Classes in the "End-User Support" package implement the behavior of the tools that are available to the users. Tools classes in the diagram are Façades (as in the Façade design pattern [**?**]) to the subsystems implementing the tools' UI and business logic. Search Services can be incrementally specified using the APIDefinitionTool, selecting the key DOM elements related to a Web search service.

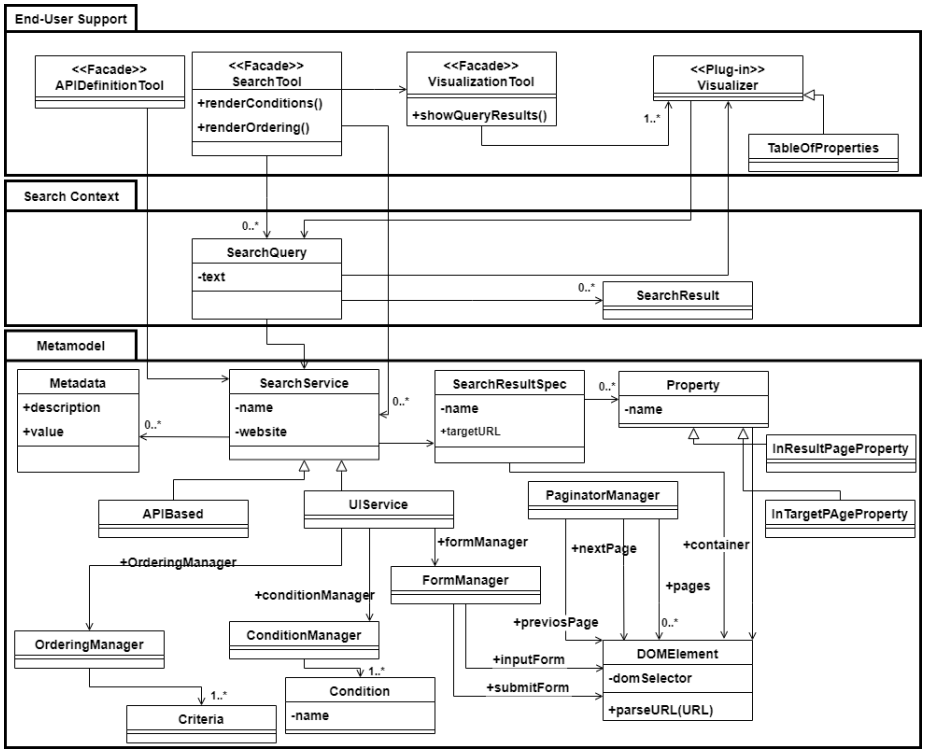

Figure 14: The design of ANDES

The SearchTool creates the corresponding menus for the existing SearchService objects and allows users to perform searches from the visited Web site. The VisualizationTool takes the search results of a search and allows users to interact with them in different ways by selecting a particular Visualizer. Visualizers (i.e., subclasses of Visualizer) implement different data visualizations (e.g., a table, charts, etc.), and presents the results considering the corresponding SearchResultSpec. Visualizers cannot be currently implemented by end-users; users customize the visualization by choosing one of the available Visualizer subclasses and changing its parameters. New subclasses, implemented by developers, can be added to the tool as plug-ins. The SearchResultSpec describes results using properties whose values are obtained from the DOM of the results page or from the DOM of the targetURL (which is the real domain object's Web page).

Users interactively select the components of the UI of an existing search engine and specify how to interpret its results. This is an end-user approach that requires no intervention from developers. A demonstration video is available online [5].

The implementation of ANDES depends on a set of UI elements present in most search services (e.g., forms, input fields, buttons). While the user interacts with the tool, as described in Section 4, instances of the metamodel classes are created.

A SearchService is the abstraction of a search engine (e.g., the search service in the NYPL Web site). For every SearchService a UISearchResultSpec must be created, together with its properties. Consider a search service in the e-commerce domain, that searches for products with a given string in its name. The search returns products, including properties such as the name, the price, and its availability. However, more specific information could be available on the site for each product, such as the technical description or its shipping costs. Both kinds of properties are extracted from different DOMs elements, which are wrapped with the Property class. Using this approach for an ancillary

[5] https://youtu.be/ZTMQfciu0-0

search allows listed properties to be obtained from different Web pages, and it is transparent from the users' point of view.

To better reproduce the power of most Web search engines, we have contemplated ordering and filtering features. Some Web sites, such as DBLP, let the users filter results according to some criteria, such as "Journal only" which is accomplished by clicking on an anchor on the search page. Our model contemplates this kind of filter through the Condition and ConditionManager classes. Ordering and filtering functionality, which depends on the original search service, is available as an action button in the visualization tool. Most search engines we analyzed offer ordering by clicking anchors or buttons, and these options are available from the same main menu in our Visualizers.

While most search services present a text input element where the user types the search query, not all of them implement the same strategy to initiate the search. ANDES supports three alternative mechanisms to initiate the search: 1) clicking on a trigger (e.g., a button or link) to load the page with the results; 2) clicking a DOM element but loading the results through an ajax-call; 3) listening to some input-related event (e.g., keypress, blur, change) and loading the results by an ajax-call. Specific subclasses implementing this behavior are WriteAndClickToReload, WriteAndClickForAjaxCall and WriteForAjaxCall respectively. The strategy is automatically assigned when the user interactively creates a search service, and each strategy knows if it is applicable based on the components the user has defined and the success of retrieving new results, so this is not a concern for the end-user.

6. Evaluation

The technical feasibility of this approach is demonstrated with the implementation of the tool ANDES. However, the operational feasibility depends on the ability of the end-user to use it correctly. In this section, we present the results of two user testing evaluations: the first one is a formative evaluation (Section 6.1), which was the first trial of the tool under construction. After fixing a few usability problems, we conducted a summative evaluation (Section

6.2) focused on the user's perception of the approach and the tool's usability. Whilst the formative evaluation was run at Obafemi Awolowo University (Nigeria), the summative evaluation was conducted at the National University of La Plata (Argentina). The language of Web sites used during both evaluations was English. However, we have localized the tool's language and the instructions according to the country and the users involved. Both the formative and the summative evaluations were based on a comparative study where participants were asked to perform the same tasks with and without ANDES. They did not use our extension before the evaluation, but they did watch an introduction and a demo video on the use of ANDES [6]. Two scenarios were presented to the participants, as follows:

- For the first scenario, the participants were asked to imagine the situation described in Section 3 of this article:
  - Read an article about literature from the New York Times: "Allegra Goodman: by the book";
  - Choose three books mentioned in such article; and then,
  - Search for these books in the New York Public (NYPL) Library Web sites to obtain further information.
- For the second scenario, the participants were asked to:
  - Navigate to a public Goodreads group called "Around the World in 80 Books". This group presents a ranking of books from more than 50K books,
  - Select 3 books from the first page of the ranking, and tell if one of those is available at the NYPL.

We followed a within-subjects protocol design in both evaluations. Half of the participants were asked to start the experiment by using the ANDES ap-

[6] Videos entitled as "Intro" in the following playlist:
https://youtube.com/playlist?list=PLHuNJBFXxaLA1FfFMtzvOXojI0yg4WWxj

proach, and the other half started using traditional navigation. If using ANDES, the participants were also asked to create the corresponding search service. The criteria used to recruit the participants in both studies were the following:

- Perform, on average, more than 3.5 searches per day using (desktop) Web browsers. We choose this number according to the number of PC searches per online searcher reported by Statista at a worldwide level (103.7 in December 2013). We have divided this number into 30 days, which results in 3.5 searches per day.
- Reading proficiency in English;
- Have used Amazon to search for books;
- Have never used the NYPL Web site.

6.1. Formative evaluation

6.1.1. Participants

We run the first evaluation with 17 participants. The participants' ages ranged from 18 to 27, with an average age of 22 years old (standard deviation close to 3 years); 8 of the participants were females. All the participants were right-handed, none of them ambidextrous. All of them were undergraduate university students, mainly from Natural Sciences; just one student was from Exact Sciences. Among the participants, 4 have been using the Web for 1-2 years, 2 of them for 2-5 years and the remaining 11 said they did it for more than 5 years. On average, they navigate the Web from a desktop environment between 2 and 10 hours a day, with an average of 5 hours (standard deviation close to 3 hours). Most of them, precisely 11 participants, use a Web browser in their daily activities, while the remaining did it a few times a week; nobody used it less often (we also provided the options "a few times a month" and "a few times a year"). 15 participants used the Google Chrome browser, one of them uses Firefox and the remaining participant said he used both. It should be noted that, in this field, no default values were given; they could enter any other browser or description. Concerning Web extensions, just two participants did

not know about them; however, usage of Web extension varied: 5 participants reported using them daily, 7 weekly, 1 monthly, 1 rarely, and 2 never use them. We also ask them how many search services they use in a typical day from a desktop Web browser; their responses ranged from 1 to 5, with a median value of 1, an average of 1.8, and a standard deviation of 1.2. Finally, regarding the number of queries that they execute monthly, 11 of them said they execute 100 or more, while the remaining did it with a lower frequency.

6.1.2. Testing procedure

The minimum specification for the laptops used in the evaluation was: Intel Core i3, 500GB of Hard disk, 4GB Ram, Windows 8 operating system, and Google Chrome v. 67.0.3396.79 onwards. We provided participants with a Lubuntu image for VirtualBox 5.2.8, to make sure all the participants used the required applications under similar conditions. The participants were presented with the scenario and they were asked to solve it with the assistance of our tool and without it. We measured user satisfaction in terms of:

- "Task Level Satisfaction". Users were asked to answer the Single Ease Question (SEQ) after every single task, to assess how difficult they found it. The question was: overall, how difficult or easy was the task to complete? Answers should match one of seven points in the rating scale, where 1 means "very difficult" and 7 means "Very Easy".

- "Test Level Satisfaction". Once the scenario was completed, users were asked to fill a SUS questionnaire, just concerning the use of our tool.

We collected interesting feedback to improve our tools and produce a new version. This feedback was obtained by asking participants the following questions:

1. Did you detect any problems in the use of the application? Which ones?
2. Name three good things about the tool.
3. Name three bad things about the tool.

Table 1: Average level of perceived ease to complete each task and search mechanism with ANDES or without it (1 to 7, 7 is the best)

Condition	With ANDES	Without ANDES
Install the tool	6.12	
Execute a demo query	5.47	
Create the augmented service	5.29	
Perform ancillary searches	5.82	5.53
Compare and choose a book	6.00	5.88
Perform ancillary searches	6.24	5.71
Compare and choose a book	5.94	5.82

4. Do you have any other comments regarding the tool, the approach, or the experiment?

6.1.3. Results

The average results for each task and each search mechanism are presented in Table 1. Although the difference is small, we could appreciate that all the tasks have been better scored under our approach.

With the feedback provided by users, a new version of the tool was released. First, we improve several parts of the JavaScript code, to do better management of the asynchronous behavior intrinsically associated with the technologies used and how they consume information from the Web. With these changes, we expected to solve the inconsistency found by some participants regarding wrong or empty results for a given query. We also discovered that the lack of precision mentioned above was related to a bug in the URL building. We also fixed a persistence problem that prevented the creation of more than one service.

Among the good things about our tool, participants said it was fast to the kind of tasks and consistent and with a good UI design. Among the bad things, some participants said that for some queries, the search results lacked precision, while other participants (5 of them) said that they could not create the search service because the tool behaved inconsistently. We obtained an average of 51.32 SUS score, which is low, so we analyzed the written feedback to improve our tool for the next stage of the evaluation process.

6.2. Summative evaluation

6.2.1. Participants

A total of 10 people participated in this evaluation, which constitutes a good minimum sample size for comparing applications in summative testing [**?**]. At the beginning of the experiment, we asked the participants to fill an initial questionnaire with some personal data (without identifying them by name or asking them for sensitive data) and some data about their technical knowledge. Regarding their nationality, 60% of them were Argentineans while the remaining were from Ecuador. Their ages ranged from 21 to 49, with an average age of 34 years old and a standard deviation of 8 years. Only one of the participants was a female; only one participant was ambidextrous.

6.2.2. Testing procedure

First of all, the participants were asked about their experience in the use of Web applications. In this point, we made them the following questions:

1. How often do you use the Web? All of them replied "daily". The other 4 options were available but never chosen: "sometimes a week/month/year" and "never".
2. How many years have you been using the Web? All of them replied "for more than 5 years". Other options were "for less than a year", "for 1 or 2 years" and "for less than 5 years".
3. How often do you use a Web browser in your daily activities? All of them said "daily".
4. Do you know Web browser extensions? All of them replied "yes".
5. How often do you use extensions for Web browsers in general? 3 participants answered "daily" while other 3 participants answered "never", 2 participants answered "sometimes a week", and the remaining 2 answered "sometimes a month".
6. Which desktop Web browser do you use more often? 6 participants replied "Google Chrome", 2 used "Safari", 1 used "Chromium" and the remaining one used "Firefox".

7. How many hours do you surf the Web through a "desktop Web browser" in a typical day? These values ranged from 3 to 12 hours, with an average of 7 hours and a standard deviation of 3.
8. How many search services do you think you use in a typical day from a "desktop Web browser"? Their responses ranged from 1 to 6 services, with an average of 3 and a standard deviation of 2.

The evaluation was carried out as follows: we gave each participant a printed copy with a description of the scenario and the required tasks (without giving details of how to do it) with both conditions (using ANDES or using the mechanisms available in the browser only). When using our approach and the tool ANDES, participants were asked to perform an extra task, which is the creation of a search service for the NYPL Web site. Participants were trained in the use of ANDES by showing them a demonstration. The experiment followed a within-subject design. During each step, the start time was registered as soon as the participant started reading the instruction for that step in the printed copy. The end time was recorded at the end of each task after they wrote down which results they obtained and chose. At the end of the full experience, we asked users to complete a SUS questionnaire for both search mechanisms, and we also ask them the same set of questions that we used for the first run of the experiment. In this evaluation, we measured usability dimensions using the following variables:

- Efficiency, in terms of "average task completion time". Once the instructions are understood, the execution time of the task is taken using the two toolsets. This includes just users who completed the task successfully and below the maximum expected time to finish the task.

- Effectiveness, in terms of "tasks completion rate". Each task was rated as "pass" or "fail". Then, the number of "pass" tasks was divided by the number of total tasks.

- User satisfaction, in terms of "Test Level Satisfaction". Users were asked

to fill a SUS questionnaire and a Satisfaction form.

The independent variable in our study concerns the search mechanisms available to perform an ancillary search, which means two conditions:

1. Using the mechanism provided by the tool ANDES;
2. Using only the Web browser and the on-site search service available on the Web site.

6.2.3. Results

All participants were able to complete all the requested tasks with both mechanisms. Table 2 presents the completion times corresponding to each participant and each search task, while table 3 provides statistical summaries. To solve the first task, searching from the New York Times article, the participants using our tool had to define the Augmented Service for the NYPL search service. Although all the participants were used to daily search in the Web and none of them have used our extension before, we did not observe a difference in the execution time of the task; in both cases, the task was performed in an average time of 9 minutes with a standard deviation of 6 minutes each. A substantial difference can be perceived in the execution time of the second task –Search from GoodReads–: the participants using our tool carried out the experience in an average time of 4 minutes and with a standard deviation of 1 minute, while the ones using the browser's integrated mechanisms did it in an average of 6 minutes and with a standard deviation of 4 minutes.

Regarding the usability measurement, the second version of the ANDES tool obtained an average SUS score of 71 points. According to the classification of Bangor et al. (2009), the SUS score of our tool (71) can be qualified as "good"; meanwhile, the SUS score for the use of the browser only to perform the task was 64.25, with can be qualified as "marginal". Regarding user performance, 15 depicts the findings. It is worthy of notice that:

- The time to solve the tasks with the browser increases with the user's age (0.48); it took more time for the older people to solve the tasks.

Table 2: Participant's execution times by task and scenario.

Participant	Search mechanism	Started with	NYT	Good Reads
1	Andes	Andes	00:13	00:03
	Browser		00:05	00:03
2	Andes	Browser	00:05	00:02
	Browser		00:16	00:14
3	Andes	Browser	00:05	00:07
	Browser		00:15	00:08
4	Andes	Andes	00:09	00:03
	Browser		00:03	00:06
5	Andes	Browser	00:24	00:07
	Browser		00:20	00:15
6	Andes	Andes	00:06	00:04
	Browser		00:11	00:04
7	Andes	Browser	00:04	00:02
	Browser		00:06	00:02
8	Andes	Browser	00:04	00:04
	Browser		00:04	00:02
9	Andes	Andes	00:08	00:05
	Browser		00:03	00:02
10	Andes	Browser	00:14	00:05
	Browser		00:09	00:06

Table 3: Statistical summary of completion times

		NYT	Good Reads
Condition 1: with ANDES	average	00:09	00:04
	STDV	00:06	00:01
Condition 2: browser only	average	00:09	00:06
	STDV	00:06	00:04

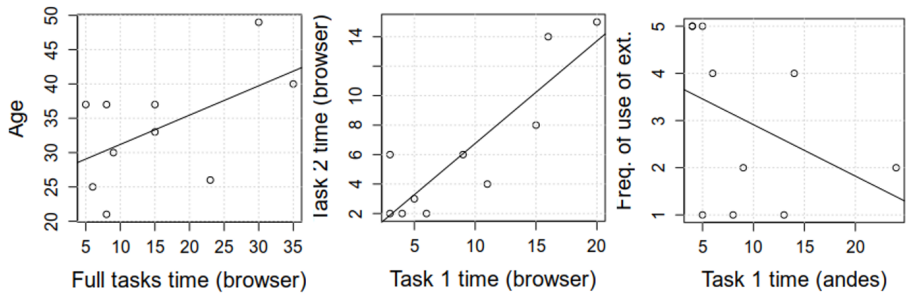

Figure 15: Correlated variables with the user's performance

- The time to complete the first task using only the browser tends to increase along with the time to complete the second task with the same tool (0.76); that is, people solving the first task in a bigger/smaller amount of time also solved the second one in a bigger/smaller amount of time respectively.

- The time to solve the first task with Andes tends to increase when the frequency of use of desktop-browsers' extensions decreases (-0.52); it took less time for people used to browser extensions.

We also collected optional feedback through a questionnaire, reported below:

- Regarding problems on the use of the tool, 2 participants said they had a problem while choosing the right container for the results. Three participants mentioned that they would like to minimize the result's popups, so they could continue executing other searches before finally comparing the results of all of them. Among the participants, 4 considered that the retrieval of the results was slow, and two of them had problems running simultaneous searches.

- We asked about 3 good things and 3 bad things about the tool. The three outstanding positive aspects were: "genericity" (3 occurrences), "fast search" (3 occurrences), and "easy to use" (2 occurrences). The 3 negative aspects were: "no link to the original result" (4 occurrences), "slow retrieval of results" (2 occurrences), "just text properties in the definition" (1 occurrence).

- They were also asked where/when would they use the tool; 10 of the participants mentioned e-commerce Web sites, 3 mentioned scientific articles Web sites, and there was a single mention of Wikipedia articles and news Web sites.

- Lastly, 3 participants said that they would like to change the visualization of the data, two users mentioned they would like to have composed services retrieving results from different sources and presenting them in the same popup, and a single user said he would like to be able to search for images with our tool.

This feedback made us aware of the need for improvements of the tool, with special attention to the interactions for the definition of search services, and the performance for retrieving results. Besides, although we included in the experiment the definition of services to demonstrate their viability, it is possible that many users can be dispensed with this task by creating and consuming shared definitions from a public repository. In this way, the definition of the search service could be carried out by one user while the rest of them use it, avoiding the extra time required for the abstraction and creation of the services.

6.3. Threats to validity

One of the limitations of this study is that giving the formative and summative nature of the two studies, the results cannot be compared. Moreover, the populations that took part in the experiments are culturally different. Although the version of the tool delivered to the participants was in their native language (either English or Spanish), the Web site used for creating the search services deliver content in English only. We recognize that the users' previous experiences with search services might affect the results. However, in the present protocol, we did not analyze how frequently users search for information using global services or on-site search services.

As we made it clear, the same scenarios of Web search were run twice. The first run, a formative study, was used and reported accordingly with our

expectations, i.e. task completion and satisfaction using some questionnaires, to obtain valuable feedback to improve our tool. This allowed us to fix bugs and a few usability issues related to the tools, which resulted in a second release of the tools that were then used for the second run of the experiment. However, although we did not obtain performance data from the preliminary study, what we can say is that a big majority of the participants were able to create the search services and use them. Between the two studies (although with different tool versions) 27 participants could use the tools for both tasks: create and trigger search services with ANDES. This is a very important result given that creating the search service is a core task in our approach.

Regarding the summative experiment, we collected the time required by the participants to perform the intended tasks. In this regard, we can see that the time required for performing an ancillary search is slightly lower with our approach, in comparison with the traditional way to do it with the Web browser. It is also important to mention that several participants said that the performance of the tool for retrieving results could be improved. We have afterward detected some points in the way the tool currently performs the ancillary search that could improve the time required to perform this task. One important improvement is to automatically pre-fetch the Web page for performing the search as well as the results from the next page (in multi-page results), which would result in a faster search. We observed that the order in which participants started the experiment did not impact the results.

7. Related works

Our work tackles the problem of information seeking in the specific case of ancillary searches. It explores new kinds of interactions with Web browsers, following an end-user development approach. In this section, we discuss related works in several dimensions. First, we analyze existing literature related to our work's conceptual aspect, which is the method used to obtain the desired information. Then we compare our proposal with other approaches that present Web browser extensions for improving the task of information search.

Finally, we discuss the relation of this work with existing end-user development approaches.

In recent years Exploratory Search has gained importance since it is not only intended to seek information by the traditional methods. Instead, this task includes more complex aspects such as cognitive activities associated with the acquisition of knowledge and the development of intellectual skills [**?**], where the main goal is to get accurate search results [**?**]. In this sense, our tool helps search for information that supports the creation of more complex searches, contributing even to the development of cognitive skills in the end user and supporting the user to get better results in the shortest possible time.

Focus + context [**?**] is a technique for visualizing information by maintaining the object of primary interest for the user presented in detail, while other contextual information is available in less detail, serving as the context for the primary object. In this way, the user can visualize both kinds of information he needs (overview/context and detailed/focus), but their presentations are different. This strategy can be associated with the idea behind inside-in ancillary searches [**?**]. In ANDES, the results of queries for extra information to solve the primary task are presented in the same context where the user needs it. Therefore, the articulatory distance between the tasks is reduced (avoiding the need to navigate between tabs to find and compare the results presented in tabs). The tool presented in [**?**] provides end-users with the means for executing ancillary searches on Web sites. However, developers must implement a broker at server-side to retrieve external and already existing services. Consequently, end-users depend on developers.

Schneiderman also presents an inside-in approach for visualizing complementary information [**?**], under the principle of "Overview first, zoom and filter, then details-on-demand". Furthermore, the author argues about the different ways of visualizing the information; in our approach, we propose that the end-user defines how he wants to visualize the information.

As [**?**] mentions, when the information is presented in overlay mode, there are better connections between primary and ancillary search, being these perish-

able and explicit, also shows that users expressed to prefer the display of results in a way that does not change their context. The authors propose implementing a solution with 2 modules; one module on the server-side and another on the client-site. The server-side module has a broker that is implemented with pre-programmed queries that can be changed, but a programmer is needed to make these changes [**?**]. In contrast, we do not need anything on the server-side, and we even provide a friendly tool on the client-side for allowing the end-user to define the structure and semantics of the results.

The studies conducted by Capra [**?**] analyzing the effects of augmenting text-based surrogates with images extracted from the underlying Webpage, conclude that very small improvements in judgment were obtained when augmenting surrogates with images. This gives support to our decision of not to include images in the ancillary searches. However, our approach supports new kinds of visualizations that can be added to ANDES as plugins.

The creation of Web browser extensions to improve Web search is not new; almost twenty years ago, the first approaches started to appear. Although, most of them are still focused on primary searches or context-aware searches, such as the case of SearchPad [**?**]. SearchPanel [**?**] is an approach based on Web browser extensibility that helps users improve their search activity. However, SearchPanel is focused on improving information seeking by augmenting results on Web search services like Google or DuckDuckGo. However, it does not allow users to create new search services specific to a domain.

In 2017, Aldalur et al. identified Web Augmentation as a promising technology for end-user development [**?**], some of these approaches are based on Web browser extensibility mechanisms to augment Web sites. For instance, Reform system [**?**] allows to attach enhancements to any Web site UI. CoScripter [**?**] empowers end-users to create their applications through Programming by Example (PBE) [**?**]. User actions in CoScripter are performed in a Web browser and saved in scripts that can be replayed and shared with other end-users, although CoScripter can be used to automate the interaction required to perform a search in a Web site search service. In the same way, Wong et al. present

a tool called Marmite [**?**], which lets end-users create so-called mashups that re-purpose and combine existing Web content and services. Another tool is NaturalMash, which enables non-professional users without any knowledge of programming and skills to create interactive [**?**] and useful mashups. All of these approaches served as a reference to develop ANDES.

It is interesting to mention that the improvement of information seeking has been also tackled from the point of view of mash-up applications and end-user environments [**?**]. Ghiani et al. [**?**] proposed a mash-up environment based on the Web browser to select widgets from existing Web pages. Our work is based on similar interaction techniques such as highlight DOM objects, components selection and executing actions to search information, which are very common in EUD environments for Web augmentation and mashups, but we differ in the interactions used. For instance, we have a wizard to abstract the components and DOM elements from web sites, the results presentation is done on the same page and our tool, unlike Ghiani's, is client-side based and running on a Web browser.

A very close work to ANDES is Excore, an approach for augmenting Web sites with automatic Web searches [**?**]. However, it is focused on a specific integration of content between two existing Web applications, and it does not support ancillary searches as a task that occurs in opportunistic ways, based on the current navigational context.

There are some articles about how to extract attributes from different types of data sources for specific purposes. Pasca et al. [**?**] extract attributes from query streams, query sessions and query logs. Fang [**?**] contributes with the extraction from Web texts, DOM trees and knowledge bases (KBs), all these extractions are carried out for the purposes of enhancing existing ontologies. Instead, we use DOM objects to build ancillary search and search APIs. Our tool lets users extract any Web site's structure without the user needing any training in Web development.

People often tend to seek information collaboratively as Gene mentions [**?**], one of those simple ways is the use of sending links through e-mails [**?**]. Our

tool also allows the end-user to store locally the definition of a search service, which can be exported and imported by the end users and thus collaborate with their colleagues sharing this information search mechanism.

It is worth noticing that our approach was meant to support ancillary search from Web browsers running on desktops. Regarding mobile Web browsers, the interaction should be re-designed. In this context, Bento Browser [**?**] offers a much better solution for search tasks than common Web browsers. Bento browser also attempts to reduce tab switching, which is a problem in this kind of device.

8. Conclusions

Search is one of the most frequent actions users perform in Web browsers. However, as we showed in this paper, Web browsers do not give good support for ancillary searches. The diversity of Web applications and users makes it almost impossible for Web browsers to be integrated with all possible search services available on the Web. This fact raises the first challenge for integration: the customization of Web browsers. A second challenge is the lack of standardized search services components among Web applications and the lack of search services APIs, impacting the automatic integration for such search services in Web browsers. The third challenge is related to the varied way in which users interact with the results of an ancillary search.

Considering all these challenges, we proposed a way to adapt the Web browser that allows end-users to customize which search services they use for ancillary searches. To give a solution for the second problem, we propose an end-user development tool to convert any Web search service (offered by a Web site) into a search service that we integrate with the Web browser. Additionally, we presented an end-user tool for triggering ancillary searches and visualizing search results in the current context. This tool also lets users customize the visualization of results, which supports the variability introduced by the third problem. It is important to remark that although we believe that desktop Web browsers are the most suitable environment to define ANDES search services,

these could be developed and used in mobile Web browsers, as other Web augmentation approaches [**?**].

Although there are some approaches to augment Web sites with information retrieved from external information sources, we think that this kind of functionality should be better integrated into Web browsers and driven by end-users. Our approach provides a visual and form-based end-user tool that allows the creation of Search Services depending on any search form's UI components. The tool uses such components to emulate the interactions required to perform searches in existing third-party search services. Such a process is transparent to the end-user and can be performed from any page and at any time. The results of the search are extracted and presented to the user in a floating panel. After the extraction, the search results are wrapped as domain objects, according to the semantic information provided during the service definition process. This semantic information will serve in the future to offer more specialized visualizations. To summarizing, all of the above contributions allow the creation of new ways of interaction and visualization of results in the context of the user's primary task.

We have carried out two evaluations. First, we have conducted a formative user study with 17 people at the Obafemi Awolowo University, Nigeria. It showed that end-users' definition of Search Service is doable and that their use makes the tasks related to ancillary searches faster. We also collected some qualitative feedback for the improvement of our tools. With a revised version of our approach and tools, we have conducted a formal experiment.

The applicability of the tool has two dimensions: i) compatibility of the ANDES extension with Web browsers, ii) compatibility with Web sites search engines. Regarding the first one, our tool is based on the up-to-date guidelines for developing this kind of software, which makes it usable in Chrome (Google Chrome or Chromium) and Mozilla Firefox. Regarding the second one, we have defined a metamodel that covers the most common aspects of the user interfaces that Web sites offer for their search engines, and we have detected several strategies for triggering the search and obtaining the results. Although

it is not possible to ensure that our tool is usable on any Web site, we can say that we used it in more than 20 Web sites that were selected from the Alexa Top Ranking.

Future work involves better exploiting the definition of metadata of the created ANDES search services and the impact of type-based or domain-specific visualizations. We also plan to extend and adapt our Webextension UI and interactions to be run on mobile browsers and evaluating its usage on small devices. Last but not least, we are considering applying machine learning techniques to automate the creation of search services that, at this point in our approach, are manually created by users.

9. Declaration of Competing Interests

The authors wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

10. Funding

This project was partially funded by the National University of La Plata, Argentina; and by CONICET, Argentina .