Chapter 14:
Building Search Computing Applications

Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Francesco Corcoglioniti, and Nicola Gatti

Politecnico di Milano, Dipartimento di Elettronica ed Informazione,
V. Ponzio 34/5, 20133 Milano, Italy
{alessandro.bozzon, marco.brambilla, stefano.ceri,
nicola.gatti}@ polimi.it,
francesco.corcoglioniti@gmail.com

Abstract. Search Computing aims at opening the Web to a new class of search applications, by offering enhanced expressive and computational power. The success of Search Computing, as of any technical advance, will be measured by its impact upon the search industry and market, and this in turn will be highly influenced by reactions of Web users and developers. It is too early to anticipate such reactions – as the technology is still "under construction" – but this chapter attempts a first identification of the possible future players in the development of Search Computing applications, by grossly identifying the roles of "data source publishers" and of "application developers", and by discussing how classical advertising-based models may support the new applications. This chapter also describes the high-level design of the prototyping environment that is currently under development and how the design will support the deployment upon high performance architectures. Finally, we describe advertising as the prevalent business model of the search engines industry, and briefly discuss the options for the evolution of such model in the context of Search Computing.

Keywords: Search Computing, software engineering, development process, advertising models, cloud computing, software architectures.

1 Introduction

The distinguishing feature of Search Computing is the ability of combining, at query execution time, knowledge extracted from various domain-expert Web sources, thus yielding to knowledge that is more accurate and complete than the knowledge available to general-purpose search systems. Such expertise (about cultural events, medical specializations, popular rock songs, and so on) is contributed through either social processes (e.g., rating, tagging, commenting) or a long and careful knowledge construction process by experts. At the current state of the art, multi-domain queries over such engines can be answered only by patient and expert users, whose strategy is to interact with specialized engines one at a time, and feed the result of one search in input to another one, reconstructing answers in their mind.

S. Ceri and M. Brambilla (Eds.): Search Computing, LNCS 5950, pp. 268–290, 2010.

With the advent of service computing and the growing interest for the Web as the predominant interface for any human activity, we expect such knowledge to become more and more exposed in the form of search services. But the mere composition of such services by sequential invocation will not solve multi-domain queries, as their interplay usually requires a lot of expertise, especially in handing and composing the search results. This challenged us in thinking to a new technology, built upon five pillars (ad hoc service definition, query optimization framework, ranking methods for join results, execution engine, and liquid queries) that collectively resolve the technical issues of Search Computing.

In this chapter, we analyze Search Computing from a broader, usage- and business-oriented perspective by addressing Search Computing applications. A *Search Computing application* is a vertical Web search application that leverages on the SeCo framework for enabling multi-domain search capabilities. The application concretely resides on a SeCo installation and consists of a configuration of one or more multi-domain queries over the existing service repository.

The chapter is made up with four main contributions. First, Section 2 presents the roles involved in the development of SeCo applications and the development process; subsequently, Section 3 describes the SeCo development environment, comprising a set of tools that support the users in their activities. Section 4 describes the SeCo reference architecture, which has been designed with the objective of being extensible, portable, and deployable upon high-performance architectures. Finally, Section 5 discusses plausible business models that could facilitate the spreading and sustainability of SeCo applications: these include advertising models and the possibility of attracting users or developing new user communities.

These contributions provide essential ingredients for building SeCo applications, but are mutually independent; therefore they are considered in four distinct sections. Sections 3-5 also include a state-of-the-art in the respective fields.

2 Development of Search Computing Applications

In this section we identify the main roles involved in the development of Search Computing applications and we describe the development process.

2.1 User Roles

Search Computing applications involve users with several roles and expertise for their configuration and usage. In this section, we identify the set of user roles involved in the development of SeCo applications, and we clarify their responsibilities and the required skills. Some roles fall outside the strict SeCo application development process, in the sense that they work for preparing the SeCo environment, in terms of platform deployment and search service development. These roles are:

- **SeCo Experts:** they are software architects that are able to deploy and configure SeCo engines over high-performance computing systems and support SeCo publishers and application developers.

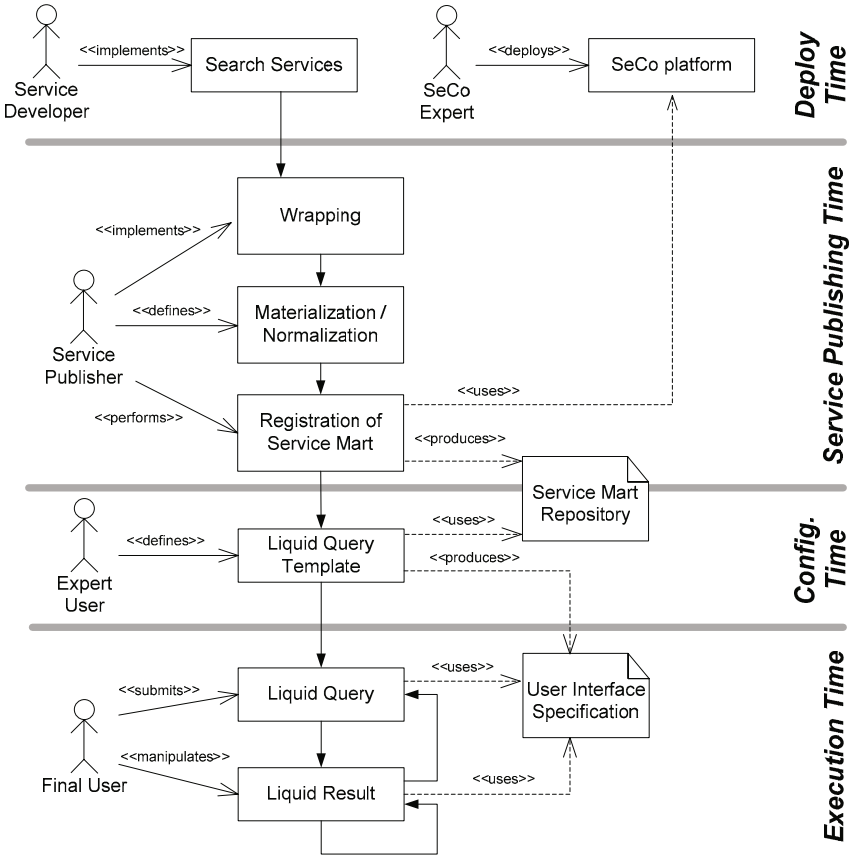

Fig. 1. Development process for SeCo applications

- **Service Developers:** they are third party software producers that publish search services on the Web. They independently produce artifacts that are needed for the SeCo applications to run correctly, but are not aware of Search services consist of any kind of REST services, SOAP services, or Web applications that can produce a ranked list of results.

Some other roles are directly involved in the SeCo application development process:

- **Service Publishers:** they are in charge of implementing mediators, wrappers, or data materialization components so as to make service interfaces compatible with the SeCo framework, and then register them within the SeCo service repository, thus defining their abstract representations in terms of service marts, access patterns, and connection patterns. Mediators adapt services that are published on the Web. Wrapping technologies span from complex wrapping tools that expose deep Web contents, to simple XSLT transformations for XML documents, to Java classes that introduce ranking and/or chunking features in services. Data

materialization tools are used to transform third party data so as to enable their publication, e.g. files, Excel sheets, or databases; these can be locally stored or acquired within the SeCo architecture, because the features of the service are too poor for granting proper treatment of the data;

- **Expert Users (or Application Developers):** they configure Search Computing application, by selecting the service marts of interest, the respective connection patterns, and associated user interfaces for query and result visualization. They also choose the complexity of the interaction interface, in terms of controls and configurability choices to be left to the user. At runtime, they interact with applications at a high level of sophistication, by composing queries on-the-fly and by executing them.
- **Final Users:** they use SeCo engines to navigate query/result interfaces devised by expert users. They interact by submitting queries, reading results, and refining/evolving them according to the liquid query philosophy.

The peculiar features of SeCo applications require new roles with respect to traditional application development. The most prominent ones are service publishers and expert users. Due to their novelty, no widespread user communities currently exist for these roles; however, they are crucial for the success of SeCo, and therefore some actions must be taken to foster the flourishing of such communities, providing them with suitable methods and tools, as we will discuss in the remainder of the chapter.

2.2 Development Process

The development process (shown in Fig. 1) is split into four main development steps:

- **Deployment Time:** this phase consists in the actual development of search services and the deployment of the SeCo platform on the suitable infrastructure. The service development and deployment is delegated to external developers and is conceptually independent from any subsequent step within the SeCo framework. The deployment of the SeCo platform as well is assigned to SeCo experts and is performed once and for all, independently on the number of actual SeCo applications that will run upon it;
- **Service Publishing Time:** several activities are needed for publishing search services within the SeCo framework: definition of the service wrappers, possible specification of materialization design for the retrieved data, normalization of the data, and registration as service marts in the SeCo service repository;
- **Application Configuration Time:** this phase, in charge to the Expert User, consists in selecting the Service marts of interests and the corresponding details, such as the connection patterns, the parameters of interest, and so on. Subsequently, the expert user defines a liquid query template for a specific SeCo application, which entails the specification of the user interface aspects. In particular, the expert user defines the structure of the liquid queries in terms of query templates that will be completed at runtime by the end user. A liquid query template is composed of:

1. a set of service interfaces;
2. a set of connection patterns for joining the involved service interfaces;
3. a set of additional selection or join predicates;
4. a default ranking function defined over the scores of service interfaces;
5. a set of possible sorting, grouping, and clustering attributes that can be applied on the extracted result set;
6. a set of positive integer values *K* that represent the possible sizes for the result pages;
7. a set of available query expansions, defined next.

- **Application Execution Time:** in this last phase, the Final User can navigate the application, i.e., the queries and results, and possibly applies some configuration details. At runtime, the end user is presented with a Liquid query Template that he can fill in with the actual query parameter. In addition, several parameters of the query template, which are initially set to defaults, can be tuned; these include:

  1. the projection attributes that define the information visible to the user;
  2. the ordering of services and of their attributes within the query;
  3. the choice of the weights of the scores of service interfaces involved in the query;
  4. the choice of cluster attributes to be used to initially visualize the query results;
  5. the optional grouping attribute to be used to initially group the query results;
  6. the choice about the size and production (e.g., continuous or chunked) of results in the result page.

Since Liquid Query vision is towards continuous evolution, manipulation, and extension of queries and results, according to the "search as a process" paradigm, the query lifecycle consists of iterations of the steps of **query submission,** when the end user submits an initial liquid query; **query execution,** producing a liquid result that is provided to the user interface; and **result browsing**, when the result can be read and manipulated through appropriate interaction primitives, which update either the liquid result or the liquid query. Depending on the kind of user interaction, the query execution performed by the engine might be suspended, restarted, or stopped. If the interaction only involves reshaping of available data, the engine may not be involved in the needed actions and the information is manipulated at user interface level.

The development process takes into account the trend towards empowerment of the user, as witnessed in the field of Web mashups (see Chapter 5 and [10]). Indeed, only the basic tasks that deal with service development (performed by service developers) require actual programming expertise. All the other design activities are moved to service registration time and to application configuration time, so that designers only need a conceptual understanding of services and queries, and do not need to perform low-level programming.

3 Development Tools

Several peculiar aspects affect the development process and the needed tools for SeCo applications:

- The **need for components provided by third parties** (in particular: search services): this implies that the process includes the need of scouting and investigating about the ecosystem of existing services within the domains of interest, for publishing and registering the found services.
- The **vertical focus** of SeCo applications: starting from the repository of available Search Services, canned interfaces can be devised for implementing verticals requiring specific domains, whose services are made available in a rich number.
- The need for **configurability of the applications**: the continuous evolution of several pieces of the architecture (services, tags and descriptions, interfaces, results) makes several steps of the development more conveniently located at query deployment time instead of service registration time.

As highlighted in the development process in Section 2.2, these features push **towards empowerment of the user** and ask for specific tools for supporting the developers. In this section we discuss the features of the existing web development tools, we highlight which of them can be borrowed for SeCo and we describe our vision towards instrumentation of the SeCo development process.

3.1 Web Design Tools and Environments

In the context of web application design, developers and designers usually exploit commercial or open source tools for performing their job. In this section we identify the classes of tools that are currently in use for Web application design, considering three main dimensions:

- **Target Users:** analysts, developers, and visual designers;
- **Development Focus:** database-oriented, service-oriented, user interface-oriented, and search-oriented;
- **Tool Availability:** local or remote.

3.1.1 Target Users

With respect to target users, we identify three main approaches, which correspond to the respective user roles:

- **Analysts and Designers:** this user role typically works with **Model-driven design tools**. Such tools include BPM (business process modeling) tools, Web engineering tools based on conceptual models, UML design tools, and MDD/MDA based techniques. Notable BPM tools that provide good Web deployment features include Oracle BPM[1], WebRatio BPM[2], and BillFish BPM[3]. The most known representative of Web engineering tools that exploit conceptual modeling and formalized development process is WebRatio[4], while a good choice of UML modeling and partial code generation (also for the web) exists. Among

[1] http://www.oracle.com/us/technologies/bpm/index.htm
[2] http://www.webratio.com/
[3] http://www.billfishsoftware.com/
[4] http://www.webratio.com/

them, we can mention MagicDraw[5], IBM Rational[6], and others. These tools provide a visual design environment that allows drawing conceptual models of the application, to debug and apply some validation, and to generate running code. Coverage of the various aspects of the application design and completeness of code generation depend on the tool. Typically, UML tools generate stub classes corresponding to the design and then provide hooks to IDEs for completing the implementation. Some BPM tools provide automatic generation of the running prototype of the web application, while more sophisticated model-driven tools like WebRatio provide refined modeling primitives that allow going for full code generation of the final application. Fig. 1 shows the WebRatio interface for designing the Web application hypertext and the contextual menu that allows the user to immediately see the generated Web application page corresponding to the selected modeling concept. The features that can be borrowed for SeCo tools include: *visual composition* of the applications (e.g., at service registration time for mapping to existing service marts; at application configuration time for selecting the marts of interest and composing them) and *automatic deployment* of the running prototype.

- **Software Developers and Debuggers:** this roles work with **Code-driven development**. This paradigm collects IDEs (Integrated Development Environment) that are explicitly targeted to web development or that covers general-purpose development but include some features or plugins addressing web issues. This class includes a set of diverse products, spanning from Eclipse WTP project[7], which provides a set of Eclipse plugins for Web applications development, to Microsoft Visual Studio. The main features that can be borrowed for SeCo tools are: *code-level support* for building and debugging service wrappers and the *code-level refinement* of the application through code inspection (e.g., for configuration files).
- **Visual Designers:** this role works with an **interface-driven approach**, developing Web interfaces with attention to detailed graphical appearance. Examples of tools that support this development approach include authoring tools, solutions like Adobe DreamWeaver[8], Aptana Studio[9], and a plethora of commercial and freeware HTML editors. As an example of interface, Fig. 3 shows the command panels of Dreamweaver. Further examples are described in the next section. The main feature that can be borrowed for SeCo tools is the support for *graphics and interface customization*, e.g., for complying with customers' visual identities.

A separate category is represented by the mashup development tools, which are of high relevance for SeCo. This category is not analyzed here because it has been widely addressed in Chapter 5. The main feature that can be borrowed for SeCo is the *online availability* of the design tools.

---

[5] http://www.magicdraw.com/
[6] http://www-01.ibm.com/software/awdtools/developer/rose/index.html
[7] http://www.eclipse.org/webtools/
[8] http://www.adobe.com/products/dreamweaver/
[9] http://www.aptana.org/

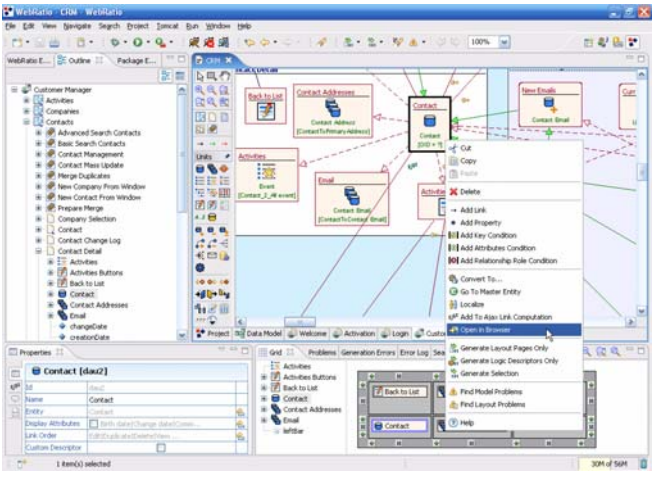

Fig. 2. WebRatio modeling interface and link to the generated Web page

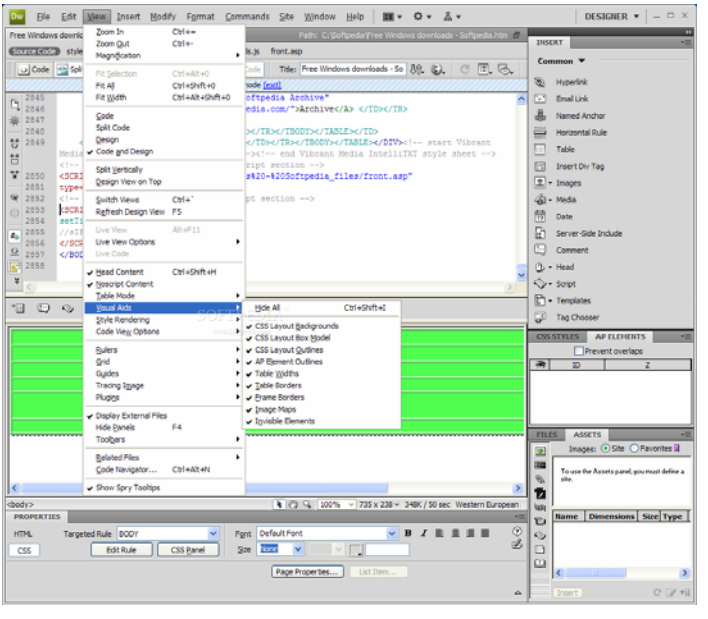

Fig. 3. Adobe Dreamweaver CS4, HTML design interface

3.1.2 Development Focus

Most of the development tools for the Web start from a specific perspective to the problem. **Interface- and interaction-oriented tools** root into the hypermedia field; they include tools like Adobe CS4[10] and Adobe Flex[11], which deliver high quality animations, interfaces, and rich applications.

[10] `http://www.adobe.com/products/creativesuite/`
[11] `http://www.adobe.com/products/flex/`

Database-oriented tools start from the opposite point of view, by allowing the design of Web applications upon published data sources. Such tools include Caspio Bridge[12], WyaWorks[13], Zoho Creator[14], Dabble DB[15], Trackvia[16], and several other similar solutions. They all allow to build Web applications made of forms, lists, and data details starting from a database schema or other data sources (e.g., spreadsheets, text files, and so on). They typically provide application templates for popular needs too (e.g., CRM, accounting, project management, and so on). Current trends move towards full-fledged online database platforms that allow publishing and management of online data sources. Most of them provide online development interfaces and Software as a Service business models. The main similarity to SeCo is the schema-based definition of services and results, as well as the structured specification of search queries.

Finally, **service-oriented tools** consider services (instead of data sources) as first class citizens for the web application. This class comprises Web service orchestration tools, mashup tools, and service repositories and registration tools. The former can be classified into two main subcategories: service orchestration tools, like Oracle SOA[17] suite (comprising a BPEL process manager, a service bus, business rules and code editors), Oracle WebLogic[18] suite (an application server specifically targeted to Web services, formerly owned by BEA), ActiveVos[19], JOpera[20], and others, whose aim is to specify executable orchestrations of services based on BPEL; and more general tools, that can be referred to as BPM tools, including Oracle BPM (born from the Collaxa BPEL engine, acquired in 2004), IBM BPM[21], BizAgi[22], and others. These tools allow the designer to describe service interactions at a more abstract level through workflow models (for instance, based on the BPMN notation). In some sense, various SeCo features refer to this vision: the *service-based invocation* paradigm, the *collaboration* between services for achieving a common result, and the *orchestration* of the query plans for producing search results.

We don't dig into the categories of mashup and service registration tools, since they have been widely discussed in Chapter 5 and Chapter 9 respectively. An example of tool at the verge between mashups and BP specifications is JOpera, that supports quick composition and orchestration of services, as well as monitoring of execution. Fig. 4 shows a sample screenshot of the tool.

[12] http://www.caspio.com/bridge/
[13] http://www.wyaworks.com/
[14] http://creator.zoho.com/
[15] http://www.dabbledb.com/
[16] http://www.trackvia.com/
[17] http://www.oracle.com/technologies/soa/soa-suite.html
[18] http://www.oracle.com/appserver/index.html
[19] https://www.activevos.com/
[20] http://www.jopera.org/
[21] http://www-01.ibm.com/software/info/bpm/
[22] http://www.bizagi.com/

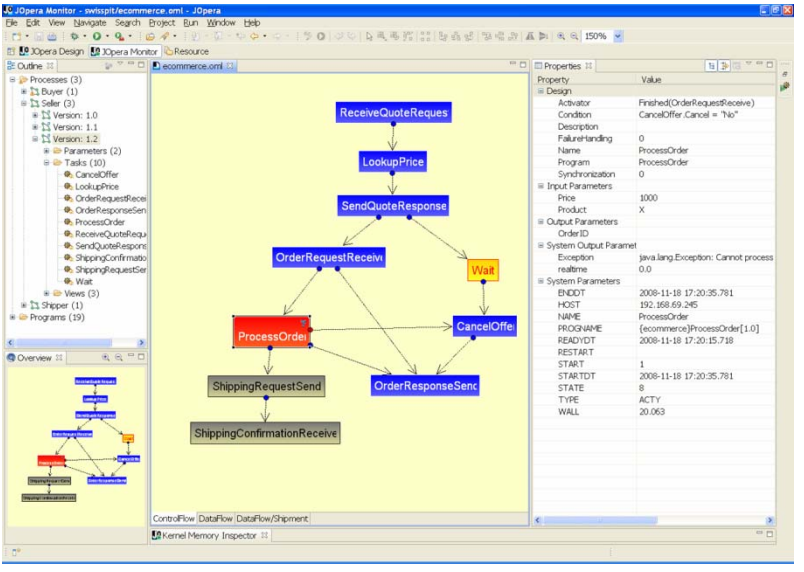

Fig. 4. JOpera Web service composition screenshot

Another emerging category of tools is related to **search-based application development**. With the increase of sophistication and the diversification of requirements that modern search solutions exhibit, the need arises of unbundling the functionality of a search system into a set of reusable components, which could be integrated to produce a variety of solutions based on the paradigm of search. One example is the Symphony platform by Microsoft, which enables non-developers to build and deploy search-driven applications that combine their data and domain expertise with content from search engines and other Web Services [23]. The similarity to SeCo is quite straightforward, although some basic features (such as join of results) are still missing in existing tools.

Other approaches to search-based development target the skilled software developer. Google Base API[23] relies on APIs for allowing developers to design their search applications. It allows to combine unstructured (i.e., full-text based) and structured (i.e., exploiting a data schema) queries and to update contents in the form of Google Data API feeds. It supports multiple ranking, overcrowding removal (thus avoiding to provide several similar items in the same result set), adjusted text results, suggestions on result schema, and much more. For example, the following query combines full-text search on digital cameras and structured search on brand "Canon":

```
snippets?q=digital+camera&bq=[brand:canon]
```

Google Base API are exposed as REST services invoked through HTTP GET, like in the following example:

```
http://www.google.com/base/feeds/snippets?bq=[brand:canon]
```

[23] `http://code.google.com/apis/base/`

Yahoo Query Language (YQL)[24], instead, allows to query, filter, and combine data from different sources across the Internet through SQL-like statements. The following YQL statement, for example, retrieves a list of cat photos from Flickr:

```
SELECT * FROM flickr.photos.search WHERE text="cat".
```

YQL is also available as a REST Service that can be invoked through HTTP GET, passing the YQL statement as a URL parameter. For instance:

```
http://query.yahooapis.com/v1/public/yql?q=SELECT * FROM
flickr.photos.search WHERE text="Cat"
```

When it processes a query, the YQL service accesses a datasource on the Internet according to a given access specification, transforms its data, and returns the results in either XML or JSON format. YQL can access several types of datasources, including Web services, REST API and Web content in formats such as HTML, XML, and RSS.

These APIs are extremely useful for SeCo, since they can be wrapped and exploited as *providers of search services*.

3.1.3 Tool Availability

A crucial aspect in modern Web application development is how development tools are made available to developers. Two major categories can be identified: tools that are **available online** with SaaS (Software as a Service) model, and tools to be **installed locally** on the developer's machine.

Among the tools available online we can mention: most mashup tools (see Chapter 5), some recent database-driven (like WyaWorks) and interface-driven design tools, the large class of CMS (Content Management System) tools, like Drupal[25] and Joomla[26], and hybrid solutions like App2You[27], which stands in between database-driven and interface-driven tools.

Desktop development tools include heavy weight solutions like Eclipse, Adobe CS4 suite, Microsoft Visual Studio[28], Webratio, and so on.

3.2 SeCo Development Tools

To comply with the SeCo vision, we foresee a set of tools to be provided to developers for covering the lifecycle phases. For SeCo application development, tools are crucial for *service registration*, *application configuration*, and *query plan tuning*, while tools for *service development* are outside the scope of the framework and interfaces for *application execution* are described in the Liquid Query approach (Chapter 13).

---

[24] http://developer.yahoo.com/yql/
[25] http://drupal.org/
[26] http://www.joomla.org/
[27] http://app2you.com/
[28] http://www.microsoft.com/visualstudio/

Service registration tools will consist of a set of facilities for allowing normalization of service interfaces and their registration as Service Marts. Tools supporting the normalization will help in:

- *Defining the Service Mart* signature;
- *Defining the Access Pattern* structures;
- *Defining the normalized schema* of the underlying data model, structured in terms of primary table and SeCondary tables as described in Chapter 9;
- *Specifying the service interfaces*, in terms of ranking, chunk, cache, and cost descriptors;
- *Defining the annotations* of the services, in terms of reference domain and keywords;
- *Establishing connection patterns* between pairs of service marts and service interfaces, to describe possible join paths for queries.

The tools will feature mapping-based interfaces that will allow picking elements from the service input/outputs (and domain descriptions) and populating the conceptual models.

Application configuration tools will allow composing application structures consisting of sets of connected service marts. The tools will support the following activities:

- *Exploring* the service repository, through visual navigation;
- *Selecting* the services of interest for the application and the respective connection patterns, including the ones needed for query expansions;
- *Defining* the interface of the query submission form and of the resultset, together with the default settings for the application and the allowed Liquid Query operations.

Query plan tuning tools will consist of a visual modeling environment that allows developers to edit query plans specified according to the Panta Rhei notation (as described in Chapter 12). Such plans are usually automatically generated by the plan optimizer, but advanced developers may want to manually refine them to take in consideration domain specific knowledge or customized choices that are not available to the optimizer.

All the tools will be developed as online applications, at the purpose of increasing SeCo application design productivity, reducing the time to deployment, and avoiding the burden of downloading and installing software.

4 Software Architecture

This section describes the architectural issues involved in the development of SeCo systems. Being SeCo a Web system dealing with a large amount of concurrent end user requests, sub-second *response time* and *scalability* are of primary importance. Therefore, high-performance architectures and deployment environments able to satisfy these requirements must be part of the solutions.

4.1 High-Performance Architectures for Web Applications

Web applications usually adopt a three-tier architecture, comprising *presentation*, *business logic*, and *data*. The data tier is usually based on a database, while in SeCo applications it consists of the registered remote services being invoked by the query engine. Scalability in Web applications can be achieved by using more powerful server machines (*vertical scalability*) or by allocating multiple server machines organized in a cluster (*horizontal scalability*)[5]. *Cluster computing* [22] enables the management of an increased traffic by splitting incoming requests to multiple servers, exploiting the fact that most user requests can be handled independently, as it happens with Search Computing queries. Different *load balancing* techniques have been devised [8] to achieve an even utilization of computation nodes. By allocating redundant nodes to replace failing machines, *failover clusters* can be used to provide *high availability* of deployed applications.

A promising deployment environment for Web applications is provided by *Cloud computing* [2] [7]. According to this paradigm, Web applications are deployed to a set of virtualized, interconnected storage and computing resources offered by third-party providers, globally referred to as a cloud to abstract from their physical location and characteristics. A cloud deployment environment (such as Amazon EC2 [1]) offers several benefits to the application provider, among which the possibilities to (1) dynamically allocate resources to an application, thus being able to dynamically scale it up to increased workloads, and (2) to eliminate fixed costs related to in-house provision of the application, paying only based on the usage of offered resources.

Short *response time* and *time-to-screen* are crucial to guarantee system responsiveness. These parameters are affected by two main factors in SeCo: internal *query processing time* and remote *services invocation time*. The former can be reduced by executing a query on multiple nodes in parallel, by exploiting *inter-query* and also *intra-query* parallelism. SeCo queries running on multiple nodes can be assimilated to *distributed queries* in a database setting [15], where a single query plan is divided into a set of sub-plans, scheduled and executed on different database nodes. However, intra-query scheduling in Search Computing is simpler (because there is no need of considering allocation of data) and can benefit from existing *scheduling algorithms* (e.g., the *work stealing* algorithm [4]) developed in the field of *grid computing* [6]. Another popular paradigm for parallel processing is *Map-Reduce* [11], a framework for efficiently distributing and scheduling computations expressed using map and reduce primitives. While Map-Reduce has proven useful for batch data processing (e.g., building a search engine index), its programming paradigm makes the execution of relational joins cumbersome; nonetheless, an extension – *Map-Reduce-Merge* [24] – has been proposed to address this issue.

Service invocation time, instead, can be reduced by minimizing and optimizing communications with services, possibly avoiding them at all. At a physical level, invocation times can be reduced by efficiently using available communication protocols. HTTP, in particular, provides facilities for *caching* Web server responses and *pipelining requests* to Web servers [12]. At an higher level, the communication problem has been addressed in *metasearch systems*, where a *crawl-metasearch hybrid approach* [9] has been proposed to reduce Web search costs, by indexing low-turnover and small data sources while meta-searching the other ones. A similar approach can be adopted for Search Computing, by recurring to *materialization* (see

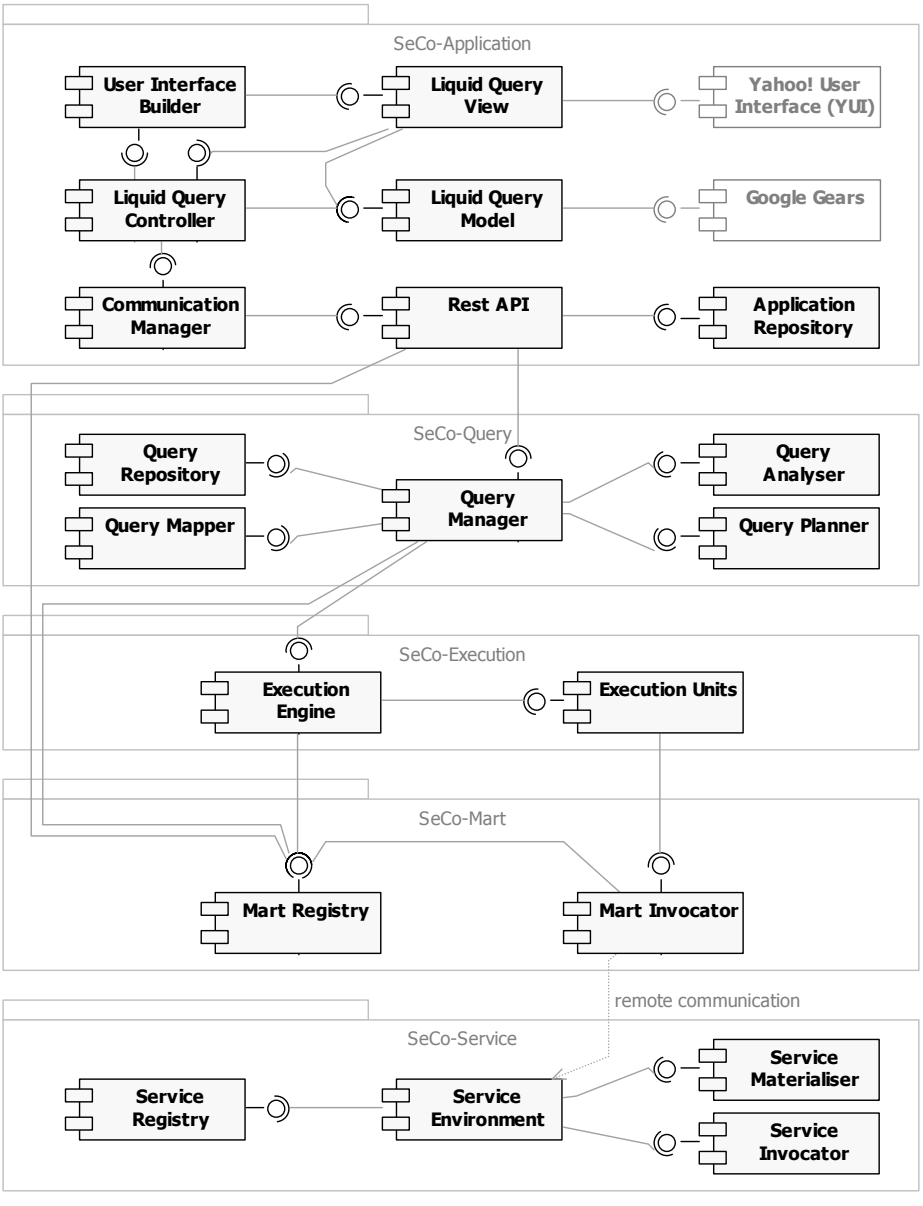

Fig. 5. UML component diagram showing the logical system architecture

Chapter 9) of frequently accessed services that provides access to small amounts of data changing infrequently. A different approach is represented by *distributed workflow systems* [20], where service nodes directly participate to the orchestration process in a peer-to-peer fashion, thus eliminating the central orchestrator bottleneck. The latter approach however is not applicable to SeCo, since it would require service providers to actively support the Search Computing framework.

4.2 SeCo Architecture

This section describes the reference software architecture designed to support the runtime execution of Search Computing queries. Fig. 5 shows the logical architecture of the system, expressed in terms of software component to be deployed (and possibly replicated) on different execution nodes. As shown in the figure, the architecture is divided in five layers:

- The lower layer, called **SeCo-Service**, is used by service developers and offers facilities to wrap and expose existing services. A *Service Registry* hold wrapper and concrete service descriptions, as described in Chapter 9. The *Service Engine* handles runtime invocations by means of a *Service Invocator* component that abstracts the service physical details.
- The **SeCo-Mart** level provides the service mart abstraction consisting of the *Service Mart Registry* and *Invocator* components, which respectively store descriptions of service marts and interfaces, and support the invocation of the latter according to the standard HTTP+JSON interface described in Chapter 9.
- The core level, called **SeCo-Execution**, contains the execution engine made of a core *Engine* component and of a set of *Execution Units* realizing the Panta Rhei model. The latter are programmed, installed, and tuned by SeCo experts.
- The **SeCo-Query** layer includes all the components required for processing a query. The *Query Mapper* decomposes natural language queries into domain-aware subqueries. The *Query Analyzer* performs the selection of access patterns service interfaces, thereby producing a service interface-level query[29]. The *Query Planner* translates the query into an optimized Panta Rhei execution plan. Queries and optimized plans are stored in a *Query Repository* for subsequent reuse, while the *Query Manager* orchestrates the whole optimization process.
- The **SeCo-Application** level provides a *Rest API* to submit queries, an *Application Repository* to store application-specific data (such as UIs' definitions) and *liquid query* support. Liquid queries are the client-side front-end of the SeCo architecture, designed as a Rich Internet Application [3] so as to enable a fluid user interaction thanks to client-side data management and to asynchronous communications with the SeCo back-end. A standard Web browser incorporates the liquid query application shell, which is an application written in JavaScript and based on a Model-View-Controller design pattern; the application leverages the libraries and functionalities offered by the Yahoo! User Interface (YUI) libraries[30] and by Google Gears[31]. The *Liquid Query Controller* initializes the application, builds the graphical user interface through the *User Interface Builder*, manages the user interactions and the interaction status, and communicates with the SeCo API through the *Communication Manager*. The *Liquid Query Model* is responsible to store and massage client data after each interaction (e.g., applying filtering sorting, aggregation of results, synchronization with a client persistent repository to enable

[29] In the current prototype, the query mapper and analyzer are not developed, as we assume that the input query is already described at the level service interfaces.
[30] `http://developer.yahoo.com/yui/`
[31] `http://gears.google.com/`

off-line usage, etc.). Finally, the *Liquid Query View* comprises the graphical objects and presentation properties specific for the SeCo applications. Client-side user interactions are associated to either *local* or *global* operations; the former can be executed directly on the client, the latter require the engine's intervention.

4.3 Deployment

This section describes the deployment of software components on processing nodes. As shown in Fig. 6, deployment is organized on three tiers:

- The **Service Tier** consists of the processing nodes providing access to registered services. A *Service Composition and Creation Framework* can be deployed to facilitate the exposing of services, and consists of the component of the *SeCo-Service* layer.

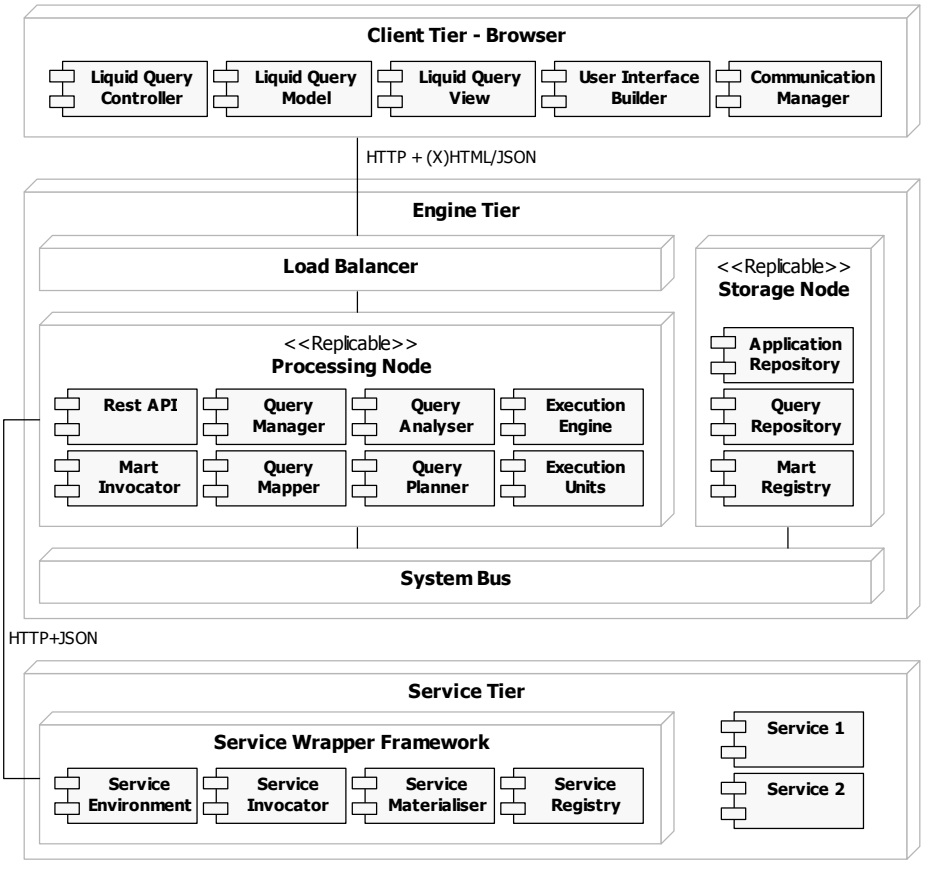

Fig. 6. Deployment of software components

- The **Client Tier** consists of client machines locally running the liquid query UI, which is offered as a JavaScript component running inside Web browsers.
- The **Engine Tier** represents the query engine, which is invoked by clients and executes Search Computing queries over registered services. Engine components can be simply deployed on a single-machine or distributed and replicated across multiple machines to achieve massive scalability. In the latter case, components can be grouped in two types of nodes, namely (1) *processing nodes*, responsible of query execution and (2) *storage nodes*, containing service and query definitions. If deployed on a cloud infrastructure, these two types of nodes can be dynamically replicated with the assistance of a load balancer, in order to cope with increasing workloads. Inter-component communication and coordination are guaranteed by a *System Bus*.

In the prototyping of the Engine Tier, besides testing functionalities, we will soon address crucial aspects such as robustness and scalability. For the second generation of prototypes, we plan to use a space-based middleware, such as GigaSpaces XAP[32], which represents a promising solution: by decoupling state (stored in space entries) from computation (provided by stateless components) it automatically supports component replication, load balancing and fail-over.

5 Business Models in Search Applications

This section discusses plausible business models that could facilitate the spreading and sustainability of SeCo applications. We start with an overview of the advertising strategies in the search field, and then we provide some hints on the possible SeCo advertising models and strategies for attracting users or developing new user communities.

The rapid growth of the Internet is transforming the way information being accessed and used. Newer and innovative models for distributing, sharing, linking, and marketing the information are appearing. As with all communication media, the major source of financial support is advertising [12]. Several Internet advertising formats are commonly used: banners, rich media, email, classifieds, referrals [21]. In today's Internet advertising industry, the so-called *search format* is the most relevant revenue-generating context: advertisers pay search engine companies to list their links (commonly called *sponsored links*) in response to specific search word or phrases. The revenue generated by the search format of advertising constitutes more than 90% of the whole revenues of search engine companies[33].

In the following three subsections, we describe the economic principles of the search format and subsequently the tools provided by the main search engine companies that can be exploited by advertisers and third parties.

[32] `http://www.gigaspaces.com/xap`

[33] About 97% of the income of Google (about 10 billion dollars per year) comes from advertising, the remaining 3% from sales of products [21].

5.1 Principles of Advertising in Search Engines

The economic principles of the web advertising search format are simple. The search engine chooses a list of sponsored links, each one composed of a head title, a brief description, and the link, to be shown (impressed) alongside the results of the search and, whenever a user clicks on a sponsored link, the corresponding advertiser [24]. This pricing scheme is commonly called pay-per-click (PPC) and is considered the fairest for search engine and advertisers. It has been shown [21] that the other two schemes, pay-per-impression (PPI) and pay-per-transaction (PPT), advantage the publisher (in this case the search engine) and the advertiser, respectively.

The idea behind the impression of sponsored links alongside search results is that a user could be interested in visiting commercial links that are strictly related to her search; this happens indeed very frequently, and therefore the revenues generated by the search format is very impressive. The choices of the list of the sponsored links to be shown and of the amount of money that a clicked advertiser must pay are accomplished by the search engine in the attempt to maximize its expected revenues, which depend on the probability that a user will click on a link and the amount of money that the corresponding advertiser would pay for that click. Obviously, the larger are such two factors, the larger the expected revenues. This problem is essentially an auction problem and is commonly studied by resorting to microeconomic tools [18].

We focus on how a search engine chooses the list of sponsored links to be shown. Given a search accomplished by a user, the first task that the search engine must address is to determine the most interesting advertisers for the user. This task is accomplished by estimating the click probabilities for each sponsored link. In doing so, the search engine exploits context information (e.g., keywords searched by the user, user's language, country, and IP) and historical data. Essentially, the click probabilities are produced by considering the last (e.g., one thousand) impressions of a sponsored link in the presence of the given context and counting the number of times it has been clicked. These probabilities are commonly called click-through-rates (CTR) and range from 0.5% to 20% with an average around 3% in practical applications.

The basic context information concerns the keywords searched by the user. An advertiser can register for one keyword or for a list of keywords, e.g., “car”, “sport car”, and “luxury sport car”; the more specific is the list of keywords, the easier and the more precise is the targeting of the advertisement to the most interested users. The advertiser can provide additional information, such as the language of the audience, the country, the region, and the city. For example, a bakery in Paris will likely target just the city of Paris, while a nationwide bank in Australia will likely want to target the entire country. The search engine will then determine whom to show a given sponsored link on the basis of several factors, including user's domain, search terms, computer's IP address (estimates its geographical location), and language preference set for the search engine.

The registration of an advertiser for a keyword (or a list) with specific language and location information is concluded by setting the maximum amount of money that the advertiser would pay when the sponsored link is clicked. This value is usually called the advertiser's *bid*. Note that such amount of money is not generally the

amount the advertiser will pay if clicked; rather, it is the largest amount that would be paid. In practical applications, the values are in the range from 0.05 Euros (minimum value acceptable by the search engine) to 15 Euros. In addition to setting such value, the advertiser can choose a maximum budget per day or a maximum number of impressions per day.

On the basis of the context, click probabilities, and advertisers' bids, the search engine chooses the list of sponsored links to be shown. Generally speaking, the search engine maximizes the cumulative revenue expected from each sponsored link. The choice of the amount of money that the clicked sponsored link must pay is an intricate technical issue, and therefore we provide only the general concepts, omitting details. On one hand, the search engine should maximize its revenue by maximizing the payments; on the other hand, it must avoid strategic behaviors of the advertisers that could decrease the search engine's profit. The aim is to produce payment rules that provide the right incentives to the advertisers to bid their true evaluations. In this way, strategic behaviors can be avoided and the economic mechanism behind the auction is said to be *incentive compatible*. The design of the most effective economic mechanism for sponsored search auction is currently an open issue in the microeconomic literature [21].

5.2 Advertising Tools in Search Engines

We review the tools provided by the three main search engines: Google, Yahoo!, and Microsoft. For reasons of space, we describe in detail the tools provided by Google and we briefly report the differences between these tools and those provided by Yahoo! and Microsoft.

Google provides several tools for Internet advertising. The basic tool for search format is AdWords [16]. This tool allows an advertiser to register for keywords, specifying language, location information for targeting audience, and upper bounds over budget and impressions. AdWords exploits GoogleMaps for the location information and can add maps to the sponsored links, as the impression of images and maps has been shown to increase the interest of users and consequently the click probability. An advertiser can also select the screen area where the sponsored link will be shown, either on the top of the search results or on the right of them, which are managed by two different auctions.

Auctions are based on the *generalized SeCond price* (GSP) [21], where the amount of money paid by the sponsored link in position i-th is the bid of the advertiser whose sponsored link appears in position $i+1$-th. In the version implemented by AdWords the price is increased by 0.01 Euro. Although this kind of auction does not produce the right incentives for advertisers to bid their true evaluations (i.e., it is not incentive compatible), it is shown to produce large revenues for the search engine and to avoid price instability in the market. Currently, Google is not interested in employing alternative economic mechanisms that in theory outperform GPS.

Google AdWords provides an advertiser with additional features: advertisers can select the *devices* and the *content networks* where her sponsored links will appear. Relative to the first feature, the advertiser can target either desktop and laptop computers, or iPhones and other mobile devices with full Internet browsers, or both. The Google Content Network [15] allows AdWords to show sponsored links also on

sites that are not search engines, including products like Google Groups and Gmail, as well as other important search sites like AOL and Ask.com, or content sites like NYtimes.com and About.com.

The tool AdSense [15] is used by website owners who wish to make money by displaying sponsored links on their websites. Website owners can use Google AdSense with two different modalities:

- The website owner can publish a Google search frame where a user can search contents through keywords. In addition to the search results, the website owner can then publish on such frame the list of sponsored links, produced by Google AdWords.
- The website owner can publish a frame wherein some sponsored links will be impressed, letting to Google AdSense the choice of best links on the basis of the content reported in the site. More precisely, AdSense analyzes the site by extracting the main keywords and subsequently submits such keywords to AdWords to produce the list of sponsored links.

With both modalities, the revenue received from the advertisements published by website owners is shared with Google. The exact ratio of the money that Google gives to the website owners depends on the specific website and is private information; usually it ranges from 40% to 50%.

Yahoo! and Microsoft provide tools very close to the ones provided by Google. Specifically, Yahoo! Search Marketing [24] is analogous to Google's AdWords and Yahoo! APT [24] is analogous to Google's AdSense. Yahoo! tools exploit the same auction model (GSP), but – differently from Google's tools - they allow advertisers to make their bids in real time. Microsoft Advertising [17] combines the services provided by Google's AdWords and AdSense. The advantages of Microsoft Advertising lay on the media network on which the advertisement can be impressed (in addition to the search engine), which contains high-traffic sites such as Facebook, Digg, Zune, and Windows Live Sharing.

All the main players (Google, Yahoo!, and Microsoft) provide sophisticated strategic tools to advertisers in order to optimize their campaigns. They allow an advertiser to monitor the number of impressions and clicks, to simulate the effects of increments ad reductions of the value of the bids, to monitor information about the users, and so on.

5.3 Business Models for Search Computing

In this section, we sketch some ideas about business models for SeCo, by explaining some scenarios for profit sharing among all players within the SeCo environment, and specifically showing the aspects of the advertising though the search format could be extended due to the new aspects of Search Computing.

A Search Computing application relies on the existence of underlying sources. A SeCo developer could decide to act independently from source owners, e.g. by using publicly available sources; then, he should play also the role of SeCo publisher and guarantee the access to the data which are needed for the application. In such a case, the business model of a SeCo application is simple, as there is only one player, the SeCo developer, whose incentive is to build an application as attractive as possible for

its perspective users. SeCo will provide to such player a new application development environment supporting a new class of Web applications, to be compared to the many environments already available.

However, the most interesting scenario for source computing is one where the SeCo developer acts as a "broker of information", by attracting content owners to participate to applications. In such a case, the business model must provide scenarios that yield to advantages both to the publisher and to the broker. Two cases are then possible:

- If the publisher gets an advantage because the traffic to the publisher's application can generate revenues for the publisher, then the model should recognize an advantage to the broker for every click to the publisher's application.
- If instead the publisher provides essential information in order for the application to become possible while having no advantage due to generated traffic, then the situation is opposite, and the model should recognize an advantage to the publisher for every click to the publisher's application.

Note that a given application might include publishers belonging to both classes. A fair model should then recognize, for every publisher/broker relationship, one of the above cases, and support it through simple contractual conditions. Advertising models can provide the underlying theory for computing the pay-per-click dues.

An interesting aspect is that SeCo applications present as results combinations of individual entries extracted from multiple services, therefore, while clicking on one link, users are choosing a "global solution" which is contributed by all other links offered within the combination. This gives rights to interesting profit sharing schemes that may give advantages also to links that, even if not clicked, contribute to a solution.

Of course, a Search Computing broker publishes a Web application, thus, as any other application, it can host search frames or frames wherein some sponsored links will be impressed, thus using the tools provided by major search companies reviewed in the previous section. Similarly, providers may open frames within their applications, and take advantage of the same mechanisms with the traffic that is carried to it through SeCo applications.

Search computing could then develop its own advertising models, and these models provide interesting problems that we plan to study, initially at a more theoretical level. Two options seem most promising:

- Multi-domain queries in SeCo may offer an important dimension for bidding, by associating bids to keywords only when other specific domains are present in a solution, e.g. a bid for the keyword "movie" only when the user is searching for "renting" in a specific "city", or instead only when the user is searching for "cinema". This option could contribute to the current development of flexible auction mechanisms within the scientific community, by adding a relevant dimension.
- A SeCo application could also act as a "broker" for sponsored links, by offering combinations which use them, and by merging the lists of sponsored links returned by multiple service providers in the attempt to rank in high position the

links that are the most appropriate for the users. In such context, the SeCo application could use a model of click probability that takes into account all the click probabilities upon the domains of the query. For instance, the SeCo application could prefer to impress links that are present in the list of all providers, rather than a link that has higher probability in one list but does not appear in the other lists.

These options are currently considered in our research so as to prepare a suitable business and advertising model for Search Computing.

6 Conclusions

This chapter aimed at broadening our perspective on Search Computing, from its enabling technologies to its architectural, usage, and business-oriented perspectives. We introduced the roles and tasks of SeCo developers and discussed how design tools can help them. We provided an overview of a reference architecture and deployment strategy, and finally we reviewed advertising models for search industry and thereby introduced the first elements of a business model for SeCo application development.

References

[1] Amazon. Elastic Compute Cloud, EC2 (2009), http://aws.amazon.com/ec2/
[2] Hayes, B.: Cloud computing. Communications of the ACM 51(7), 9–11 (2008)
[3] Farrell, J., Nezlek, G.S.: Rich Internet Applications The Next Stage of Application Development. In: 29th International Conference on Information Technology Interfaces, ITI 2007, June 25-28, pp. 413–418 (2007)
[4] Blumofe, R.D., Leiserson, C.E.: Scheduling multithreaded computations by work stealing. J. ACM 46(5), 720–748 (1999)
[5] Bondi, A.B.: Characteristics of scalability and their impact on performance. In: WOSP 2000: Proceedings of the 2nd international workshop on Software and performance, pp. 195–203. ACM, New York (2000)
[6] Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic models for resource management and scheduling in grid computing. Concurrency and Computation: Practice and Experience 14(13-15), 1507–1542 (2002)
[7] Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 25(6), 599–616 (2009)
[8] Cardellini, V., Casalicchio, E., Colajanni, M., Yu, P.S.: The state of the art in locally distributed web-server systems. ACM Comput. Surv. 34(2), 263–311 (2002)
[9] Craswell, N., Crimmins, F., Hawking, D., Moffat, A.: Performance and cost trade-offs in web search. In: ADC 2004: Proceedings of the 15th Australasian database conference, pp. 161–169. Australian Computer Society, Inc., Darlinghurst (2004)
[10] Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. IEEE Internet Computing 11(3), 59–66 (2007)

[11] Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: OSDI 2004: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, pp. 10–10. USENIX Association, Berkeley (2004)
[12] Even-Dar, E., Kearns, M., Wortman, J.: Sponsored Search with Contexts. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 312–317. Springer, Heidelberg (2007)
[13] Feng, J., Bhargava, H.K., Pennock, D.: Implementing Sponsored Search in Web Search Engines: Computational Evaluation of Alternative Mechanisms. Informs Journal on Computing (forthcoming), http://ssrn.com/abstract=721262
[14] Fielding, R., Gettys, J., Mogul, J.C., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol (1998), http:/1.1.Tech.rep.
[15] Google. AdSense (2009), https://www.google.com/adsense/
[16] Google. AdWords (2009), https://www.google.com/adwords/
[17] Kossmann, D.: The state of the art in distributed query processing. ACM Comput. Surv. 32(4), 422–469 (2000)
[18] Mas-Colell, A., Whinston, M.D., Green, J.R.: Microeconomic Theory. Oxford University Press, Oxford (1995)
[19] Microsoft. Microsoft Advertising (2009), http://advertising.microsoft.com/
[20] Muth, P., Wodtke, D., Weissenfels, J., Dittrich, A.K., Weikum, G.: From centralized workflow specification to distributed workflow execution. J. Intell. Inf. Syst. 10(2), 159–184 (1998)
[21] Narahari, Y., Garg, D., Narayanam, R., Prakash, H.: Game theoretic problems in network economics and mechanism design solutions. Springer, Berlin (2009)
[22] Pfister, G.F.: In search of clusters, 2nd edn. Prentice-Hall, Inc., Upper Saddle River (1998)
[23] Shafer, J.C., Agrawal, R., Lauw, H.W.: Symphony: Enabling Search-Driven Applications. In: USETIM (Using Search Engine Technology for Information Management) Workshop, VLDB Lyon (2009)
[24] Weber, T.A., Zheng, Z.E.: A model of search intermediaries and paid referrals. Tech. rep., 02-12-01, The Wharton School, University of Pennsylvania (2003), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=601903
[25] Yahoo! APT from Yahoo! (2009), http://apt.yahoo.com/
[26] Yahoo! SearchMarketing (2009), http://searchmarketing.yahoo.com/
[27] Yang, H.C., Dasdan, A., Hsiao, R.L., Parker, D.S.: Map-reduce-merge: simplified relational data processing on large clusters. In: SIGMOD 2007, pp. 1029–1040. ACM, New York (2007)