Symphony: A Platform for Search-Driven Applications

John C. Shafer, Rakesh Agrawal, Hady W. Lauw

Search Labs, Microsoft Research
1065 La Avenida St., Mountain View, CA 94043, U.S.A.
{jshafer, rakesha, hadylauw}@microsoft.com

Abstract— **We present the design of Symphony, a platform that enables non-developers to build and deploy a new class of search-driven applications that combine their data and domain expertise with content from search engines and other web services. The Symphony prototype has been built on top of Microsoft's Bing infrastructure. While Symphony naturally makes use of the customization capabilities exposed by Bing, its distinguishing feature is the capability it provides to the application creator to combine their proprietary data and domain expertise with content obtained from Bing. They can also integrate specialized data obtained from web services to enhance the richness of their applications. Finally, Symphony is targeted at non-developers and provides cloud services for the creation and deployment of applications.**

I. Introduction

While people world-wide now turn to a general web search engine to satisfy their general information needs, they at the same time seek specialized sites for information on specific topics. Examples include Expedia for travel, Amazon for books, eBay for rare artifacts, WebMD for health, IMDB for movies, Ticketmaster for events, etc. Google Custom Search (google.com/coop/cse) and Yahoo! BOSS (developer.yahoo.com/search/boss) represent efforts by general search engines to enable people to build custom search engines that alter the default behavior of the underlying general-purpose search engine. Such a custom search engine may, for example, restrict the search to some domains, automatically add terms to an input query, or reorder search results to give preference to some URLs.

A natural progression in this line of evolution is for a search engine to take up the role of a world-wide resource for unstructured data to be used in building and deploying search-driven applications. For instance, a video store owner may be interested in an application that allows people to browse his inventory of movies, but augments his content by integrating it with focused search results for supplemental content such as the latest reviews and trailers obtained on the fly from a search engine. A wine connoisseur may create and embed in her web site a specialized search vertical that combines her knowledge of wines with targeted web-search results, and may be able to monetize her efforts using an advertising or referral service.

We present the design of a system, christened Symphony, to assist in the development of such search-driven applications. While Symphony naturally makes use of the customization capabilities offered by the search engines (our prototype uses Bing), its distinguishing feature is the capability it provides to the creators of the search applications to be able to combine their propriety data and domain expertise with the content obtained from the search engine (and other data services). In this way, Symphony users can focus on developing and curating quality content, making the best use of their specialized interests and knowledge, while still being able to fall back on a search engine for general content. The integration of specialized data and Web resources makes richer results possible.

Another important feature of Symphony is its target audience. Rather than professional application developers, Symphony is designed to be usable by domain experts and enthusiasts, who may not have the requisite development skills to build a search application from scratch. Symphony carries out the heavy lifting, and allows its users to concentrate on functionality and design. The key capabilities provided by Symphony include provision for storage and indexing, integration of proprietary data with Web resources, easy-to-use design interface, support for monetization, and deployment of search applications to users' Web sites or social network platforms (e.g., Facebook).

II. System Design

We present here the key aspects of Symphony's design. We first describe Symphony's functionality from the perspective of an application designer and then describe a few system internals.

A. Creating a Search-Driven Application

Assume for present purposes that the application designer already possesses the proprietary content and domain expertise to be incorporated into the application. Content could be inventory data for a retailer, a list of favorite movies or wines for an enthusiast, etc.

Symphony offers the following capabilities to the designer.

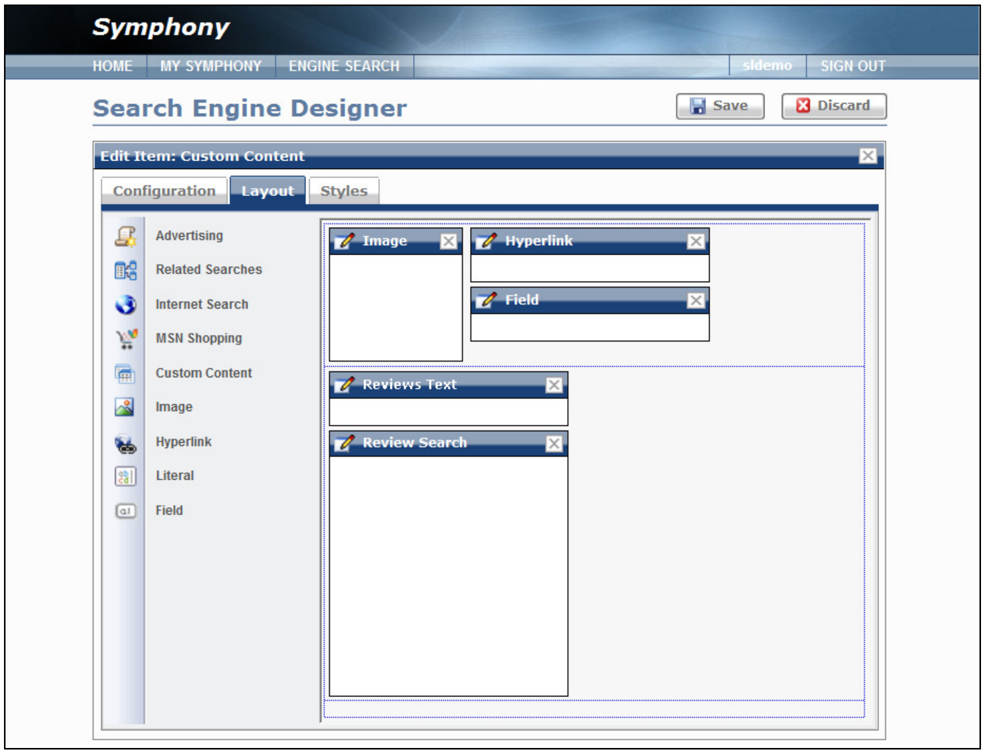

Fig. 1 Design Interface

Proprietary Data: Symphony provides private and secure space to store and index proprietary data belonging to the application designer. It supports a variety of upload methods (e.g., HTTP/FTP file upload, RSS feeds, or URL crawling), as well as a variety of structured data formats (e.g., delimited files, Excel files, and XML). Symphony also supports dynamic data accessed through SOAP and REST-based web services. This facilitates real-time data freshness, allows users to keep data considered too sensitive "in-house" and allows integration of 3rd-party services.

Built-in Services: Symphony has built-in support for accessing internet content in the form of web/image/video/news results from a search engine. Most services support additional configuration, such as site restriction. A Site Suggest [2] feature is provided that can suggest additional related sites to include based on the set already specified. We also integrate with advertising services such as adCenter (adcenter.microsoft.com), allowing ads to be displayed and configured just like any other content source.

Data Integration: The various proprietary, 3rd-party and built-in data sources can be integrated flexibly, for instance by using one data source as a primary content (which will be queried based on the application user's query) and one or more data sources as supplemental content (which will be queried based on selected fields from the primary content).

Design Interface: Symphony provides a WYSIWYG-style design interface – no coding is required. Fig. 1 shows a screenshot of the design interface implemented in our current prototype. The left bar shows various data sources that application designers can drag-n-drop onto an application, specifying how each should be searched and how many results to be shown. Multiple data sources can be added to the layout and arranged as desired. This drag-n-drop process is also used to configure how individual results should be laid out. Application designers can create HTML elements such as text, images and hyperlinks using fields from the data source. The right panel in Fig. 1 illustrates one such layout. In this case, a search result features a hyperlink, an image, and a descriptive field. In addition, a search result may trigger another Internet search over supplemental content. Supplemental content can be added by simply dragging additional data sources onto the current result layout. The designer selects which fields from the first data source to use when querying that secondary data.

Presentation: In addition to general layout organization, further customization of the application's look-and-feel is supported via templates, wizard-style assistance from Symphony, or through style properties on individual elements (e.g., color, font-size). For more web-savvy users, greater control is possible via style-sheets.

Distribution: Application designers can embed the application into their own web sites by copy-and-pasting auto-generated snippets of JavaScript and HTML onto a web page. Symphony also supports publishing the application to social networking sites such as Facebook.

Hosting: Regardless of how an application is distributed, its execution and the resources involved are always shouldered by Symphony. This facilitates seamless

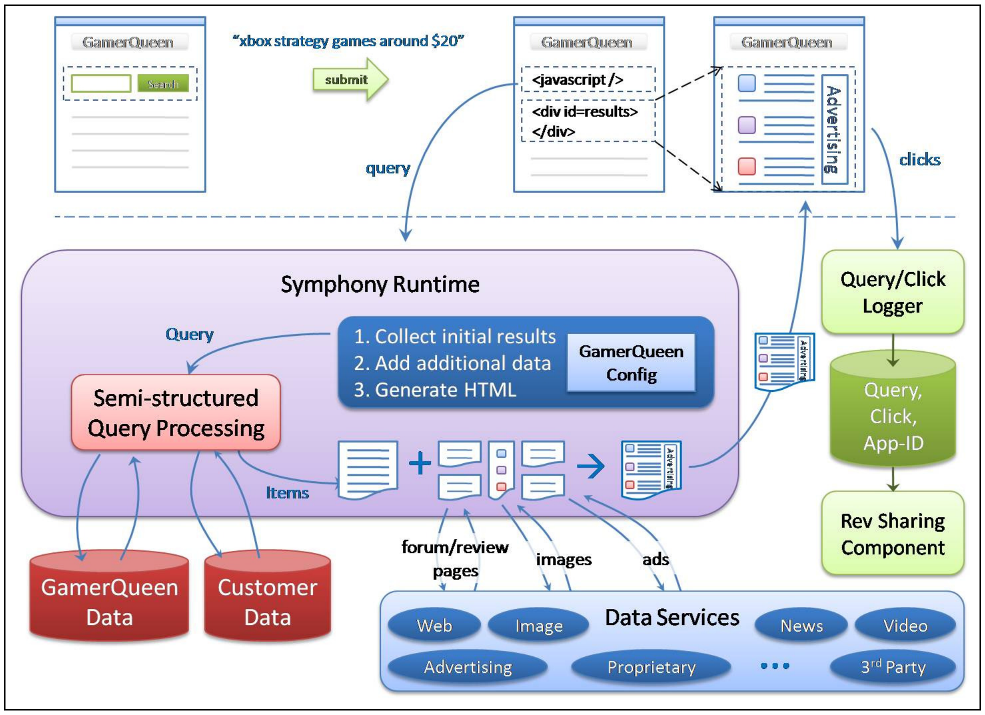

Fig. 2 Query Execution in Symphony

integration while minimizing any resource requirements on the part of the application designer.

Monetization: Symphony has built-in support for the application designer to be able to record customer interactions with the application and obtain various summaries. When a link is clicked in a Symphony-hosted application, it can be logged by the system. If the click is on an advertisement from an integrated ad service, the application designers will automatically be credited by that service for any ad-click revenue. In another scenario, a summary of an application's click traffic can be downloaded by the application designer to serve as the basis for charging or auditing referral compensation.

B. Application Example

A video game store owner, Ann, wants to create a custom search experience around her inventory of video games. After registering her proprietary inventory data with Symphony, Ann drags the inventory data onto a new application layout as a primary content, and configures the application to search by title, producer, and description. She then configures the result layout to show the title hyperlinked to a detail page, an image, and a description. Ann may then wish to include game reviews as supplemental content by dragging web-search content onto the result layout and restricting it to sites such as gamespot.com, ign.com and teamxbox.com, which she knows to be high-quality sites for game reviews. The game titles from the inventory data could then be selected to drive that web search. If Ann had a real-time pricing and in-stock service available, it too could be included as service-based supplemental content.

C. Application Execution

The key Symphony components that come into play when processing a query are highlighted in Fig. 2. We continue with the example in Section II-B, and assume that the owner, Ann, has published her application to her site, GamerQueen.

When a customer submits a query on the GamerQueen site, the auto-generated JavaScript on that page immediately forwards the query to Symphony for processing. The Symphony runtime must now process the query according to the configuration of the GamerQueen application. The query is first processed by the primary content sources, which in this scenario means searching GamerQueen's proprietary inventory data. In a more complex scenario, customer data could also be included to alter the query to, say, prefer some types of games over others.

Next, the supplemental content sources are queried based on data fields from the results from the primary content. The fields that should be used as arguments in these queries are specified by the application designer in the configuration file for the application. In this scenario, a focused web search retrieves links to game reviews for each game result. This supplemental content is then merged with the selected display fields from the GamerQueen proprietary data and formatted into HTML, applying any configured layout and presentation details. The resulting HTML is sent back to the embedded JavaScript, which then injects it into the GamerQueen page as a list of game results complete with reviews from the web.

Note that to the customer, the entire experience appears as though it occurred seamlessly within the GamerQueen web site, even though all the processing and heavy-lifting was off-loaded to Symphony.

TABLE I
COMPARISON OF SYMPHONY WITH RELATED SYSTEMS

	Symphony	**Y! BOSS**	**Rollyo**	**Eurekster**	**Google Custom**	**Google Base**
Search API	Bing	Yahoo	Yahoo	Yahoo	Google	Google
Custom Sites	Supported	Supported	Supported	Supported	Supported	No
Proprietary, Structured Data	Supports various uploads (HTTP or FTP, RSS, xls, txt, xml)	Limited to partners	No	No	No	Supports various uploads (RSS, txt, xml)
Monetization	Ads voluntary (revenue-sharing)	Ads mandatory	Show your own ads	Ads mandatory for for-profit entities.	Ads mandatory for for-profit entities.	No
Custom UI	Drag'n'drop	Mashup Python library, HTML/CSS	Only for partners	Basic styling (e.g., colors, fonts)	Basic styling (e.g., colors, fonts)	No
Deployment of Search Applications	Hosted at server, published to 3rd-party sites, or Facebook	No assistance.	Only allows search box on 3rd-party sites	Only allows search box on 3rd-party sites	3rd-party sites	Data to surface on Google's search products

III. RELATED WORK

Table I shows a comparison of Symphony with prior attempts by commercial entities to assist users in the creation of custom search engines. Yahoo! BOSS is a SDK targeted at developers to build search applications on top of Yahoo!'s infrastructure. Its offering consists of a Web API to make query calls to Yahoo!'s search index, a client-side Python library to create custom mashups, and a possible partnership allowing specific third parties to integrate their proprietary data with Yahoo!'s infrastructure. Unlike Yahoo! BOSS, Symphony goes beyond simply opening up Bing's search infrastructure to third parties. It aims at making the creation of a custom search engine as painless and seamless as possible to the average lay user, by providing additional support in the form of no-code, drag-and drop, user-friendly interface, hosting, and deployment of custom search engines. Symphony makes its monetization schemes voluntary, and shares any revenue with the designer. Rollyo (rollyo.com) and Eurekster (eurekster.com) only offer site-restriction capability with limited styling customizations such as colors or fonts.

Symphony also goes beyond systems such as Google Custom Search that simply allows a user to tweak the default behavior of a general search engine. The key difference is that a Symphony user can combine his proprietary data with Web search services to create search-driven applications. Our goal is also entirely different from GoogleBase, in that we are not looking for users to provide us with data to improve our search results. Instead, our goal is to facilitate designers to leverage their own intellectual property (e.g. proprietary data) to improve the search experience for their targeted end users.

The capability to integrate information from several data sources to produce unified search results is somewhat similar to the notion of Web mashups [1]. Several commercial systems, such as Microsoft Popfly (popfly.com) and Yahoo! Pipes (pipes.yahoo.com), assist users in the creation of mashups by offering a programmingless interface to pull together several data sources. However, unlike Symphony, they do not offer access to integration with proprietary content. Neither do they offer indexing or searching over mashed up data. There are other proposals for making mashup creation easier, for instance with, spreadsheet [3], network navigation interface [4], or scripting language [5]. However, such implementations still require some form of learning effort on the part of the users (e.g., learning a scripting language).

IV. CONCLUSIONS

We have presented an overview of the Symphony platform and shown how it can be used to create custom search applications that leverage both a user's proprietary content and his or her domain expertise. Its significance goes beyond individual search applications. As each application is usually oriented around a specific topic or community, the usage data (query and click logs) generated from various search applications may eventually provide topic- or community-specific relevance signals to the general search engine. As future work, we will explore how Symphony could further facilitate the creation of rich search applications through such means as recommending suitable supplemental content (e.g., good game review sites) for a designer's primary content (e.g., game inventory), supporting richer querying of structured data, adding support for social search features, and creating new applications by composing other applications.

REFERENCES

[1] S. Abiteboul, O. Greenshpan, and T. Milo, "Modeling the mashup space", in *Proc. WIDM'08*, 2008, p. 87-94.

[2] A. Fuxman, P. Tsaparas, A. Kannan, and R. Agrawal, "Using the wisdom of the crowds for keyword generation", in *Proc. WWW'08*, 2008, p. 61-70.

[3] W. Kongdenfha, B. Benatallah, J. Vayssière, R. Saint-Paul, and F. Casati, "Rapid development of spreadsheet-based Web mashups", in *Proc. WWW'09*, 2009, p. 851-860.

[4] B. Lu, Z. Wu, Y. Ni, G. Xie, C. Zhou, and H. Chen, "sMash: semantic-based mashup navigation for data API network", in *Proc. WWW'09*, 2009, p. 1133-1134.

[5] M. Sabbouh, J. Higginson, S. Semy, and D. Gagne, "Web mashup scripting language", in *Proc. WWW'07*, 2007, p. 1305-1306.