UDC 004.4'236

RIA AND SOA MATCHING IN ENTERPRISE WEB 2.0

Dimiter G. Velev
Department of Information Technologies and Communications
University of National and World Economy, Sofia, Bulgaria
e-mail: dvelev@unwe.acad.bg

Service Oriented Architecture (SOA) has become a standard for engineering IT systems [1, 2]. SOA helps quickly and effectively create, integrate and reuse application services so systems can be built and updated as fast as business requirements change. While SOA transforms application development and integration for IT, the benefits are often much less obvious to enterprise users. SOA applications look and feel like most other web applications and application delivery cycle improvements can be hard for the end-user to understand. The new abstraction "service" is the basis for constructing flexible applications. Thus the business logic (code) is encapsulated as "services" that can be invoked and assembled as applications to represent the business process.

Web 2.0 is the next advancement of the Web as a serious platform for building software applications [5, 6]. The term Enterprise 2.0 points to the usage of Web 2.0 technologies. Web 2.0 for an enterprise will combine collaborative behavior with transactional behavior.

Rich Internet applications (RIA) are web applications that have the features and functionality of traditional desktop applications [3, 5, 7]. RIAs typically transfer the processing necessary for the user interface to the web client but keep the bulk of the data back on the application server.

RIA refers to the development aspects of highly interactive web applications which can range from simple to very complex. Enterprises desktop applications are able to use the Web as the user interface, instead of the desktop. With RIA, enterprises can combine the best of client-server (rich user interactions) with the power of the Web. However, many web technologies from the past (DHTML, Javascript, CSS, DOM, Flash/Flex, VB) have serious limitations in terms of high-end scalability, security, reliability and programmer productivity.

Traditional web applications centered all activity around a client-server architecture with a thin client. Under this system all processing is done on the server, and the client is only used to display static content. The biggest drawback with this system is that all interaction with the application must pass through the server, which requires data to be sent to the server, the server to respond, and the page to be reloaded on the client with the response. By using a client side technology which can execute instructions on the client's computer, RIAs can bypass this slow and synchronous loop for many user interactions.

Internet standards have evolved slowly and continually over time to accommodate these techniques, so it is hard to draw a strict line between what constitutes a RIA and what does not. All RIAs share one common characteristic [5, 6]: they introduce an intermediate layer of code, often called a client engine, between the user and the server. This client engine is usually downloaded as part of the instantiation of the application, and may be supplemented by further code downloads as use of the application progresses. The client engine acts as an extension of the browser, and usually takes over responsibility for rendering the application's user interface and for server communication.

What can be done in a RIA may be limited by the capabilities of the system used on the client. Generally the client engine is programmed to perform application functions that i will enhance some aspect of the user interface, or improve its responsiveness when handling certain user interactions, compared to a standard Web browser implementation. In most RIAs the client engine performs additional asynchronous communications with servers.

Advantages of RIA

Although developing applications to run in a web browser is a much more limiting and difficult, than developing a common desktop application, there are clear advantages of adopting RIA [3, 5, 7]:

- The volume of the installed application is smaller;
- Updates to new versions can be automatic;
- Users can use the application from any Internet connected computer ;
- Many tools exist to allow off-line use of applications;
- Most RIA technologies allow the user experience to be smooth regardless of the client's operating system;
- Web-based applications are generally less prone to viruses.
- Richer functionality. RIAs can offer user-interface behaviors not obtainable using only the HTML components that are available in browser based Web applications. This richer functionality may include anything that can be implemented in the technology being used on the client side.
- Greater responsiveness. The interface behaviors are typically much more responsive than those of a standard Web browser that must always interact with a remote server.
- Client/Server balance. The demand for client and server computing resources is better balanced, so that the Web server need not be the center of operation load that it is with a traditional Web application. This frees server resources, allowing the same server hardware to handle more client sessions concurrently.
- Asynchronous communication. The client engine can interact with the server without waiting for the user to perform an interface action such as clicking on a button or link. This allows the user to view and interact with the page *asynchronously* from the client engine's communication with the server. This option allows RIA to move data between the client and the server without making the user wait.
- Network efficiency. The network traffic may also be significantly reduced because an application-specific client engine can be more intelligent than a standard Web browser when deciding what data needs to be exchanged with servers.

Disadvantages of RIA

- Enabled scripting. A scripting language is often required. If the user has disabled active scripting in their browser, the RIA may not function properly, if at all.
- Client processing speed. To achieve platform independence, some RIAs use client-side scripts written in interpreted languages such a consequential loss of performance (.
- Script download time. Although it does not have to be *installed*, the additional client-side intelligence (or *client engine*) of RIA applications needs to be delivered by the server to the client. While much of this is usually automatically cached it needs to be transferred at least once and the script download time may be long.

- Loss of integrity. Because a RIA client can modify the RIA's basic structure and override presentation and behavior, it can cause failure of the application to work properly on the client side.
- Loss of visibility to search engines. Search engines may not be able to index the text content of the application.

Enterprise Web 2.0

The goal of Enterprise Web 2.0 mission is to bring together RIA, SOA and miscellaneous systems to deliver business-critical applications over the Web [2, 3, 5]. This combination of technologies provides benefits to both the end user through an RIA front end and IT with the ability to consume services and deliver them to the end user in a business-critical capacity. The ability of Enterprise Web 2.0 to deliver business-critical applications is the result of a well-thought-out architecture that overcomes three core problems with leveraging the Web as a service platform:

- Complex and uncontrolled end-user environments
- Stable communications over HTTP
- Service consumption

General obstacles to RIA-SOA – Client model

- Lack of Web services or SOA: Web services adoption is actually still in the early stages for many organizations.
- Governance Constraints: The reuse of Web services inside a given organization is not quite an easy task as freely using any Web services found in the Web.
- Tools and Practices Are Little Known: RIA SOA/Clients are still a relatively new idea with immature tools and techniques.

The advent of RIA technologies has introduced additional complexity into Web applications which makes them harder to design, test and support [5, 6]:

- Greater complexity makes development more difficult. The ability to move code to the client gives application designers and developers far more creative freedom. Alas this in turn makes development harder, increases the high level of availability of software errors.
- RIA architecture breaks the Web page paradigm. Traditional Web applications can be viewed as a series of Web pages. RIAs disables this model, introducing additional asynchronous server communications to support a more responsive user interface.
- Asynchronous communication makes it more difficult to isolate performance problems.
- The client engine makes it more difficult to measure response time. For traditional Web applications, measurement software can reside either on the client machine or on a machine that is close to the server. The RIA architecture disables this approach, because the client engine breaks the communication between user and server into two separate cycles operating asynchronously.

It is obvious the SOA will have to change some of its basic assumptions since this has driven a search for new models. Most new Web 2.0 applications start out with an API since getting connected to other parties will help for the business growth and the focus on openness as fundamentally essential characteristics to building an internal partner is a must.

Web 2.0 is used in has two aspects predominantly —one social, the other technical [1, 5, 6]. On the social side, Web 2.0 is about a phenomenon of shifting the publishing power out to users and away from centrally controlled publishing processes. The ability for users to blog and

syndicate their posts, the notion of a wiki as a collaboration amongst users and the evolving idea of a mashup as something the user can assemble from existing Web parts and data are all examples of the power to compose being provided to the many. Web sites and Web applications using AJAX to improve ease of use make it even simpler for users to compose blogs and assemble mashups. Desktop-installed software increasingly is being displaced through the use of AJAX and services [4].

This trend [1, 3, 5] touches the RIAs, mashups, AJAX, RSS, REST and other Web 2.0 areas. Now being referred to as Enterprise 2.0, the Web 2.0 technologies are helping to create rich interactive front ends to SOA back-end systems. Besides that lot of experienced business users, who typically are nondevelopers, can take services and build mashups without IT involvement.

SOA simplifies the creation of new services and helps old services meet changing business requirements; EW2.0 needs to be able to easily accommodate changes to the services [5, 6]. EW2.0 centrally managed application deployment mechanism and zero install means that when a change is deployed to the server, all end users will immediately be updated the next time the application is invoked.

RIA is the best way to build a user interface and SOA is the best way to build and expose business services, then combined these technologies are capable of building superior applications. An application that combines RIA with SOA will benefit from the dynamic characteristics of an asynchronous, transactional and self-distributing user interface with the advantages of a loosely-coupled and reusable services layer. New architectures and frameworks will undoubtedly arise to address the security and scalability concerns of the RIA-to-SOA applications. RIA applications also must consider how to be backward compatible to the current non-RIA Internet. Even though the RIA-to-SOA relationship is still emerging, organizations and engineers should realize their power and look to them to create a new breed of applications.

Literature

1. Adam Michelson, SOA + RIA + OSS = Web 2.0, http://searchsoa.techtarget.com/
2. Appcelerator: RIA + SOA, http://ajaxian.com/archives/appcelerator-ria-soa
3. Dion Hinchcliffe, RIA clients: Not just for the Web anymore, http://blogs.zdnet.com/Hinchcliffe/?p=11
4. Lee Thé, Improve Business Productivity With AJAX and SOA, http://adtmag.com/article.aspx?id=21394
5. David Linthicum, RIA, SOA & Web 2.0 Mashups - Mash What? http://www.web2journal.com/read/532018.htm
6. Nolan Wright, The Next Web Development Episode Is RIA + SOA, http://www.sys-con.com/read/513263.htm
7. Rich Internet application,http://en.wikipedia.org/wiki/Rich_Internet_application/