

# Navigating the Next-Generation Application Architecture

**Chuck Hutchinson and Jeff Ward**, *Command Information*  
**Karen Castilon**, *Independent Consultant*

In today's Internet environment, hyperconnectivity and new and complex technologies require a shift from individualized proprietary enterprise applications to a more open and dependent world of cloud computing. This article identifies changes IT professionals must understand to make the necessary adjustments to practices in today's evolving development space.

**T**he Internet is rapidly evolving into an always-on, always-connected, device-independent environment. Enterprises are also evolving, with mobile workforces operating in dispersed and more technologically complex environments. Meanwhile, a convergence of technology trends is transforming the very nature of applications, including service-oriented architecture (SOA), software as a service (SaaS), rich Internet applications (RIAs), Web 2.0, and, encompassing them all, the concept of cloud computing. Just as meteorologists monitor cloud formations and shifting winds to forecast an impending storm's strength and course, the developer community needs to track these industry trends and understand their impact. At the 2008 Enterprise 2.0 Conference, Rishi Chandra, product manager for Google Enterprise, declared, "The next 10 years of innovations are going to be in the cloud. Enterprise software is not going away, but there is a transition taking place."<sup>1</sup>

We face significant and, in many ways, radical changes to the application development environment. This article investigates this dramatic shift in two ways: first, we survey the existing cloud computing landscape; second, we look at this new paradigm's impacts from both a technical and business perspective to better understand how organizations will need to navigate the various challenges they will face. The key factors affecting IT professionals are how technical resources' roles will change and how organizations will need to adjust the way they currently manage the development life cycle.

## Surveying the Landscape

Recent work describes cloud computing as a "paradigm in which information is permanently stored in servers on the Internet and cached temporarily on clients."<sup>2</sup> The cloud is merely a metaphor for the Internet, and thus, unlike traditional application architectures that rely on the infrastructure and services provided and controlled

within an enterprise, cloud computing uses the Internet and its emerging set of technologies and services to satisfy the computing needs of individual users and organizations.

In many ways, this concept inverts the traditional enterprise architecture. Instead of relying on controlled, internal resources, the enterprise can look outward, accessing and integrating applications and services that others have created and control. Just as a traditional client-server application uses the LAN, the SaaS model uses the cloud. The Internet infrastructure makes this possible. With the cloud as a single, ubiquitous point of shared access, business-to-business interactivity fundamentally changes.

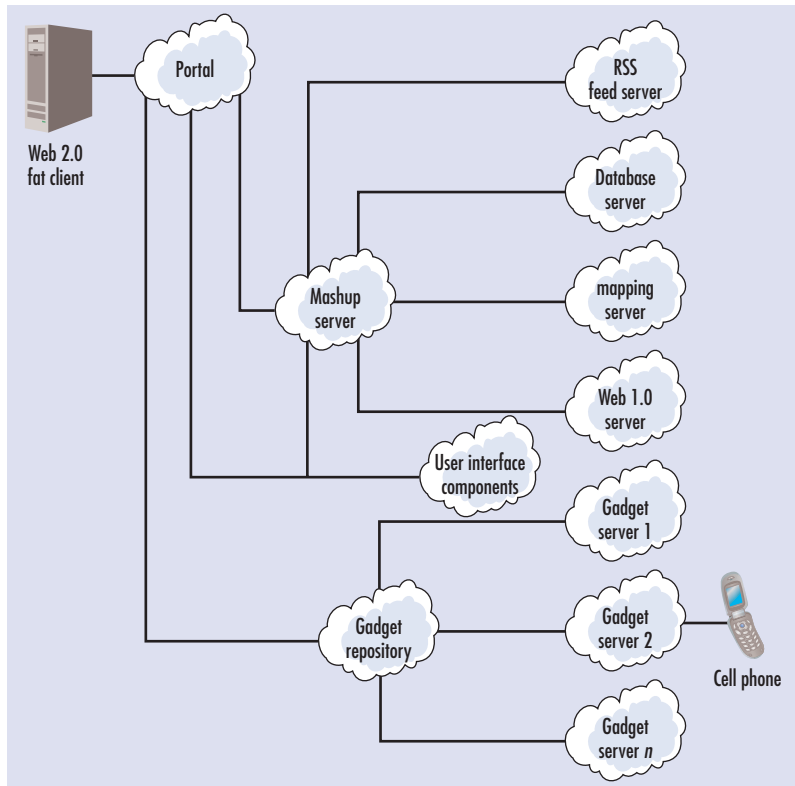
### The Next-Generation Application Architecture

Unlike the traditional application architecture's relatively static structure, this next-generation architecture is a dynamic collection of moving parts, with no clear concept of a baseline; it's far more dependent on relationships and interactions between organizations.

The cloud computing architecture has a wide variety of reusable components, including portals, mashup servers, RSS feeds, database servers, mapping servers, gadgets, user interface components, and legacy Web servers. Architects can integrate these elements to form an edge application in which multiple services and components are overlaid or "mashed" together.

Figure 1 shows this new computing model's landscape. The components' location in the diagram is significant: those further to the right are smaller in nature and more reusable; those in the center represent components that contain, aggregate, and extend services via mashup servers and portals. They provide the general user experience in a portal or Web page.

In this architecture, your application can reuse several different components—the servers can exist on the Internet or within the enterprise. Combining these components lets organizations build new, powerful applications, thereby delivering greater business value faster and at a lower cost.



**Figure 1. Cloud computing architecture. The components' location is significant: those further to the right are smaller in nature and more reusable; those in the center represent components that contain, aggregate, and extend services via mashup servers and portals.**

Most of the elements in this sample architecture are dynamic in nature, operate in a SaaS model, and leverage a SOA. In Figure 1, they provide edge data services to enable a mashup application in which the data from one service (such as addresses in a database) can be mashed up with mapping information (such as Yahoo or Google Maps) to produce an aggregated view of the information. This type of data visualization has become a common form of mashup on the Web and serves as a popular example of cloud computing. Other components in this example provide communication mechanisms to other technologies (such as SMS messages to cell phones). In some cases, gadgets or widget components provide both the user interface elements and business logic that deliver application functionality with little or no programming effort.

Surveying the cloud computing landscape, we see a variety of next-generation architecture components; it's easier to understand how they work by viewing them as categories of components, each containing a set of available elements or services that provide a specific type of capability

or serve a particular functional need. Let's review these various categories of components and their capabilities.

## User Interface Components

User interface components such as Ext.Js or Dojo provide rich user interface controls to craft high-end, fat client Web 2.0 applications. These components offer a variety of different windowing controls, such as tab controls, floating pane windows, resizing components, scroll bars, and forms that offer a rich end-user experience. Sliders, selection lists, and grids enable the design of Web 2.0 data entry forms, with data entry validation performed at either the client or server side. Ajax requests are made from these user interface components to issue HTTP posts to access different edge services.

Your application can also obtain these user interface components directly off the cloud at runtime—for example, the AOL Developer Network hosts numerous APIs for application use and several versions of Dojo (0.4.2, 0.4.3, 0.9.0, 1.0.0, 1.0.2, 1.1.0, and 1.1.1) for edge applications.

## Reporting Components

Reporting components, such as fusion charts, provide users with new visual representations of data, allowing high-end dashboard applications that give brief yet qualitative organizational assessments. Bar charts, speed gauges, and other controls can also aggregate data and provide high-level charts that give a real-time snapshot.

## Portal Components

JSR-168/286 portal components, such as Life-ray, allow applications from a variety of different mashups, widgets, and iframes to be included in different user panels. Portals also let users build applications through simple customizations in which the user specifies a URL for a panel or pane. Web 2.0 users can customize their own interfaces, reusing the components they need without programming. This new deployment model gives end users more control over their applications.

## Server-Side Components

Server-side components, such as JackBe's Presto, provide mashups and mashlets that enable the development of mini applications to access databases, RSS feeds, Lightweight Directory Access

Protocol authentication, and Web clipping (also called HTML parsing). High-end development tools aid in service generation; some mashup servers provide tools that generate data transfer objects to access databases.

Mashup servers also let customizations encapsulate services. Mashup servers are similar in nature to application servers such as J2EE containers, but they offer a wider variety of customization options—for example, developers can craft XPATH expressions to clip Web pages, write JavaScript customizations to inject business logic, and play with other high-end filters to manipulate business data. Mashup servers can also provide for publishing Microsoft Excel spreadsheets, letting business users customize their applications with off-the-shelf tools. Some high-end mashup technologies even let business users craft new services by joining, merging, filtering, and mashing them up with other services, which lets users build the application's business logic with point-and-click, drag-and-drop, and cut-and-paste technologies. This is a significant point: business users can now control the integration of their business services without the cost associated with high-end application development.

Gadget repositories act as brokers in which enterprises can register their services for others to use. Within the repository, developers can search for different SaaS components and, in some cases, select from several similar options. In others, developers can get HTML script tags to embed the component within their applications. Google Gadgets provides a higher level of reusability, in which developers can reuse services from other applications in new ones. Developers can also write gadget interfaces to other technologies, letting components bridge technologies—for example, Google Gadgets can send SMS messages from the Web to cell phones.

Mapping services allow for the customization of localized services mashed up with enterprise data. A new set of enterprise applications could enable a new view of the same data set—for example, some real estate foreclosure applications map foreclosed properties, allowing institutions to evaluate an area to understand its mortgage risks. Both Google and Yahoo mapping technologies let developers build their own widgets by instantiating a GMap2 and several GMarkers to build custom applications. Other mapping

controls allow for scaling, overview, continuous zoom, scroll wheel zoom, and traffic overlays to manipulate the map.

Other applications use business listings—for example, Yahoo’s Local Search Service component lets users query for different types of businesses in the radius of a given latitude and longitude. Taking that information and mashing it up with a Google or Yahoo map gives the user the ability to map restaurants, retail stores, schools, and so on, as well as services to geocode an address (that is, take an address and convert it to latitude and longitude for future map plotting).

## Navigating the Storm

With this growing list of available Internet-based services and components, it’s easy to see the impact that cloud computing is having on the nature of applications. However nascent it might seem, though, this next-generation architecture signals a significant departure from traditional development, and the developer community will need to adapt its current stable of tools, skills, and development practices.

## Development and Testing Tools

What development tools exist for this next-generation environment? Developers can use Firebug, a Firefox plug-in, to inspect CSS elements, JavaScript code, rendered HTML, the Document Object Model (DOM) rendered within the browser, and network transactions made from the page to download JavaScript or issue Ajax requests. Firebug also lets developers set breakpoints and step through JavaScript code.

Microsoft provides a similar tool, the IE Development Toolbar, which is integrated into Internet Explorer. Using the IE Development Toolbar, developers can monitor the DOM, inspect the CSS, and highlight and monitor the HTML rendered within the browser.

JSLint is a Web-based JavaScript syntax checker that developers can use for JavaScript verification. It’s similar to the `lint` application for the C programming language that assesses C programming syntax errors. With JSLint, developers paste their JavaScript code into the syntax panel and hit the JSLint button. The site then reviews the JavaScript code and returns any errors or warnings. It detects some of JavaScript’s most common problems, such as missing semicolons

and commas as well as undeclared objects and functions. Developers can also configure JSLint to ignore different warning types.

JsUnit is a unit-testing framework similar to JUnit or NUnit. To use it, developers code test suites that include one or more HTML test pages, each containing one or more JsUnit tests. Within each JsUnit test case, the developer writes assertions assessing the code’s behavior. If those assertions fail, then the test fails. JsUnit also provides support for setup and teardown methods to deal with the construction and cleanup of objects required for testing.

JsUnit also provides a Web-based test runner to execute JsUnit test suites. This test runner executes the tests and notes their successes and failures. It also provides support for a tracing window to log information during testing.

This next-generation architecture signals a significant departure from traditional development, and the developer community will need to adapt.

## Technical Concerns

As always, working in a new environment presents new and different challenges for developers. Several SaaS services require the developer to provide a token or key at runtime to represent the organization using the application. Both the Google and Yahoo mapping APIs, for example, require developers to register their applications’ URLs. During the registration process, both the Google and Yahoo interfaces generate a unique key that the developer must provide at runtime to Yahoo or Google to retrieve the mapping JavaScript. Another SaaS issue with which developers must be aware is how a component’s interface might change over time. The addition of new methods or properties typically isn’t an issue, but removing a method or property can result in runtime exceptions, and the application might fail to behave as expected.

This latter concern raises a new issue with cloud-based applications: not having complete control over all application components means you must rely on an external entity for component quality and reliability. This in turn presents

a whole new range of concerns during application deployment: Was the component tested? How adequate were the tests? Does the controlling entity have servers that can scale to manage the demand? How will the demand of one application affect the demand on another? Do you need a service-level agreement with this external party? Will you incur some cost for using its services? How mature is this organization? Are its components open source? Can you get support with development problems?

Other issues stem from how the components are delivered to the end user. In a client-server architecture, the Web 2.0 client is a fat client—this browser has support for JavaScript and other scripting techniques, but it must also act as a container to issue requests from multiple edge services simultaneously. Some services provide JavaScript components that developers must customize to build applications. At runtime, your application will download these components off the cloud to the client. Depending on the type and version of the browser, the results can vary. Sometimes an application might work in Firefox but not in Internet Explorer; other times, an application might work in Internet Explorer 6 but not in Internet Explorer 7. Organizational browser requirements could indeed dictate how an application needs to be developed.

From a development viewpoint, unit testing in this environment is problematic. Due to the nature of cloud computing, in which applications consume Internet-based services, it's difficult to perform unit testing; developers often must run full integration tests. They also need to consider the environment in which the components are deployed and what impacts that might have—for example, proxy servers could be required to pass the HTTP requests through.

Within the enterprise, not all data is readily accessible. There aren't yet enough Web services in existence to meet every need—either in the enterprise or on the Web itself. Additionally, governance and security are always a concern when accessing enterprise data.

## Business Concerns

Organizational security policies will inevitably raise governance and authentication issues within an enterprise. Some SaaS components require Internet access, but an organization might have

a security requirement to use proxy servers for all external Internet access. To comply with the enterprise security profile, architectural components such as mashup servers and RSS feed requests must be configurable to pass these requests through proxy servers.

When developing applications in this environment, employing an agile methodology helps mitigate risk. It makes sense to develop next-generation components in isolation and then integrate them into the application after they're working properly.

**A**s architectures continue to evolve, it's critical for IT professionals to remain at the cutting edge of technology. Interdependencies on external data sources will become even more far-reaching than they are today, so it's also important to continue to participate in open communities for discussion and education. This type of collaboration ensures that we won't be left behind in the ever-changing world of application development. ■

## References

1. M. Ricciuti, "Google's Enterprise Vision is in the Cloud," *CNet News*, 10 June 2008; [http://news.cnet.com/8301-10784\\_3-9964654-7.html](http://news.cnet.com/8301-10784_3-9964654-7.html).
2. C. Hewitt, "Perfect Disruption: The Paradigm Shift from Mental Agents to ORGs," *IEEE Internet Computing*, vol. 13, no. 1, 2009, pp. 90–93.

**Chuck Hutchinson** is a senior consultant at Command Information. His interests include IPv6 multicast programming, cloud computing, and mashup technologies. Hutchinson has an MS in software engineering from George Mason University. Contact him at [chuckhutchinson2@yahoo.com](mailto:chuckhutchinson2@yahoo.com).

**Jeff Ward** is a principal with Command Information. His interests include next-generation applications and the generational impact on technology, as well as Web site and intranet design, IT strategy consulting, and the agile methodology. Ward has a BS in aerospace engineering from the University of Maryland. Contact him at [jeffw66@hotmail.com](mailto:jeffw66@hotmail.com).

**Karen Castilon** is an independent consultant whose interests include enterprise mashups, online privacy and safety, and digital media. She has a BA in business management and economics from North Carolina State University. Contact her at [kcastilon@verizon.net](mailto:kcastilon@verizon.net).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.