Develop lightweight ancillary tools connected to electronic health records systems

Build your own solutions on top of existing EHR systems

Shahid N Shah
CEO and Chief Architect
Netspective Communications, LLC

Skill Level: Intermediate

Date: 14 Dec 2010

The 2009 stimulus bill's HITECH Act offers money to physicians, hospitals, and multi-hospital systems if they become "meaningful users" of certified electronic health records (EHR) systems. In order to meet *meaningful use* rules for EHR systems, you will need to create your own architecture to support lightweight applications. This article describes how to add meaningful use and other business requirements as lightweight ancillary tools connected to your EHR system without adding risk to your existing environment.

Meaningful use has catapulted EHR back into the limelight

The 2009 stimulus bill's HITECH Act offers as little as tens of thousands of dollars to private physician practices to as much as tens of millions of dollars to hospitals and multi-hospital systems if they become "meaningful users" of certified EHR systems. The government has done a good job defining *meaningful use* (MU) to be technology and vendor-agnostic. With the National Institutes of Standards and Technology (NIST) spearheading the testing and certification criteria, the requirements for meeting MU are clear. See Resources for more information about meaningful use and the U.S. government incentive programs.

Now that MU criteria and test plans are available, developers with single, small applications will be able to update their applications within a few months to a year and be ready for certification. The current certification guidance works well for independent software vendors that sell to multiple customers; however, it's not as convenient or easy for legacy vendors, open source firms, or those that build their own in-house medical records systems.

In order to meet MU rules for in-house EHR systems or systems from more well-established commercial vendors, such as Meditech who provides EHR systems

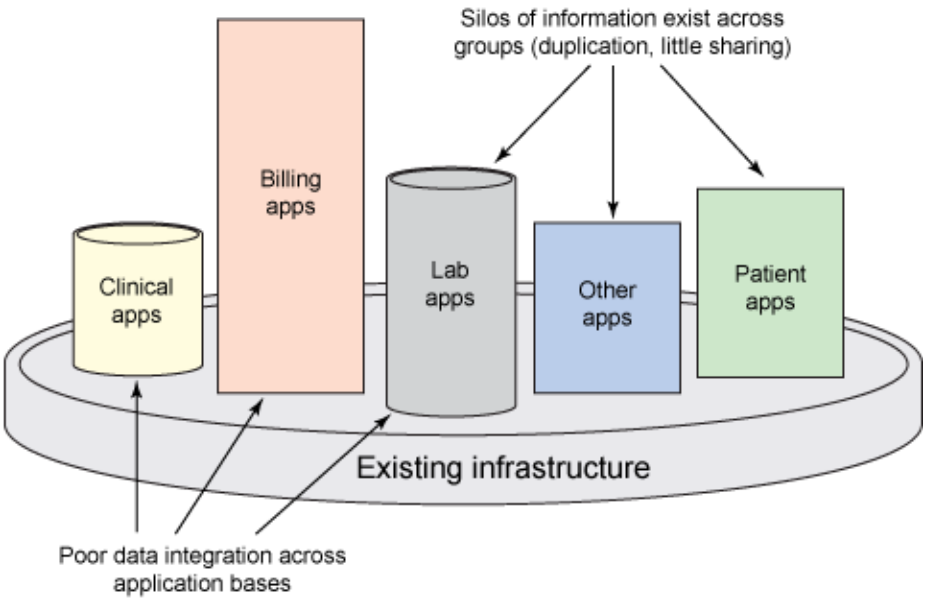
to many hospitals, you will need to create your own architecture to support the new functionality. This paper describes an approach that allows you to add MU requirements as lightweight ancillary tools connected to your EHR system.

The monolithic single-vendor approach won't work long-term

Many CIOs support a monolithic or a single-vendor approach to EHR systems because they believe it simplifies their environment, gives them one contact when there are problems, and ultimately reduces IT risk. Hospital IT systems are more complex than average systems. While the single-vendor, monolithic application is nice in theory, most CIOs reluctantly understand that they cannot live with one vendor for their EHR and meet all of the hospital end user goals and all regulatory requirements, including meaningful use. Vendors do not like to update their systems because of one customer; they add functionality and services to their systems when there are many customers that need the same new functionality. The vendor's feature road map will rarely align directly with the hospital IT strategy and requirements, so the hospital CIO will want other options.

Another reason that single-vendor approaches may be difficult is that successful vendors often grow by acquisition, not organically. They purchase separate systems from various companies and need to integrate them into the hospital IT environment. The typical healthcare user must interact with a number of different applications, even if the applications come from the same vendor. For example, a private practice physician must interact with applications for scheduling, managing an electronic medical record, writing orders, dictating a transcription, reviewing claims submissions and patient e-mails, and continuing medical education. A nurse will have to maintain the schedule, handle the ICD/CPT coding and billing, perform the intake and out-processing of patients, answer phone calls, conduct medical record management and follow-ups, and participate in continuing education. It is obvious that these jobs and responsibilities require information to cross the boundaries of each of these applications, even if they come from the same vendor. Each time users have to move from one application to another (or from a paper-based way of doing things to the computer), time is wasted re-entering information or retrieving duplicate information (see Figure 1).

Figure 1. Inefficient existing infrastructure



Service-oriented architecture (SOA) and an integrated approach is necessary

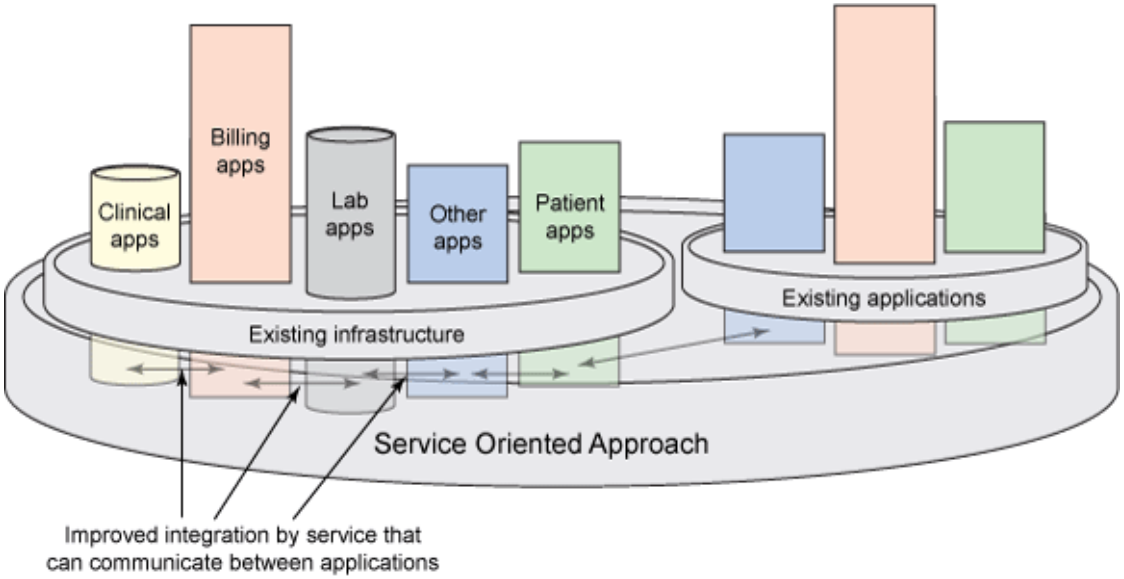
Legacy EHR systems were created with an application-focused mentality. To integrate data and keep from duplicating functionality and information capture we need a way to model and create common services instead of applications. In this context, *services* refers to pieces of computer code that are small, reusable, and focused granules of functionality instead of large monolithic applications. For example, an application used to register a patient or edit patient demographics would use a common *patient registration service*. An application used to discover the prescriptions for a patient would use a common *patient medications service*. An application that accesses a care provider's credentials would use a centralized *credentialing service*. These are all examples of healthcare services that were developed once and then reused across dozens of applications.

Safety-critical systems used at the point of care or within medical devices have special considerations in both practical and regulatory terms. For example, a system that tracks patient vitals in real-time for health monitoring in an IV pump requires significantly more attention to design, construct, test, and validate than one tracking billing records for insurance reimbursement. The service-orientation approach allows safety-critical systems that may be used in regulated medical devices to be designed and tested differently than pure information-focused systems.

By defining and designing specialized services, and allowing small applications to wrap the reusable services, applications become less monolithic. Smaller and more nimble services are easier to adapt to changing business and regulatory requirements. Considering that every application requires something as common as patient registration functionality, a patient registration service allows new applications

to be built faster and with better quality. Extend the example to other common services and multiply the cost savings across dozens of applications and the reductions of time and costs become significant enough to affect the bottom line in most IT groups. Freeing up precious IT resources to focus on additional business optimization, allows even organizations outside IT to see top line growth due to innovation (see Figure 2).

Figure 2. The SOA approach



Lightweight applications wrapped around services are needed now

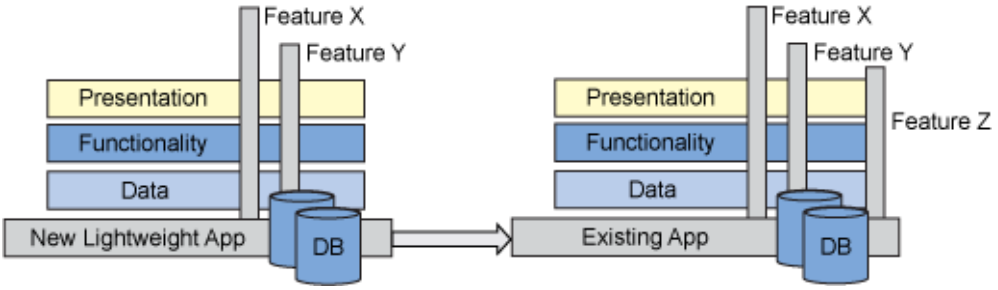
If you are responsible for MU strategy and implementation at your hospital or care provider organization you're going to run into practical challenges. You probably have monolithic applications already, but you know that you need more service orientation. You need to figure out how to create lightweight applications that you can wrap around existing services, or how the applications built on top of existing vendor applications and can directly access the databases. Neither is easy but you have four primary options:

- Copy functionality between applications (not recommended).
- Integrate data but copy functionality.
- Integrate data, functionality, and business logic.
- Integrate user interface, functionality, data, and business logic.

Option 1: Copy features and enhance (everything is separate)

Option 1 is the typical legacy model where you copy features and enhance them, and where all new code is separately written, tested, and deployed (see Figure 3). This is the most expensive way of developing software and causes significant duplication of effort. This method is not recommended.

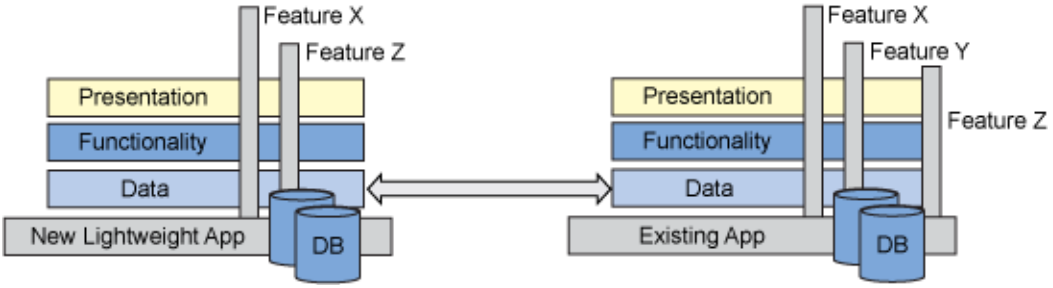
Figure 3. Option 1



Option 2: Integrate existing data, but copy features and enhance

The second option is the most common integration approach when you have no access to the application code but you can get to the database. You can tie data from the new application to data from the existing application by accessing the existing database from the new application (see Figure 4). This approach is recommended if you cannot use the more sophisticated options laid out in options 3 and 4.

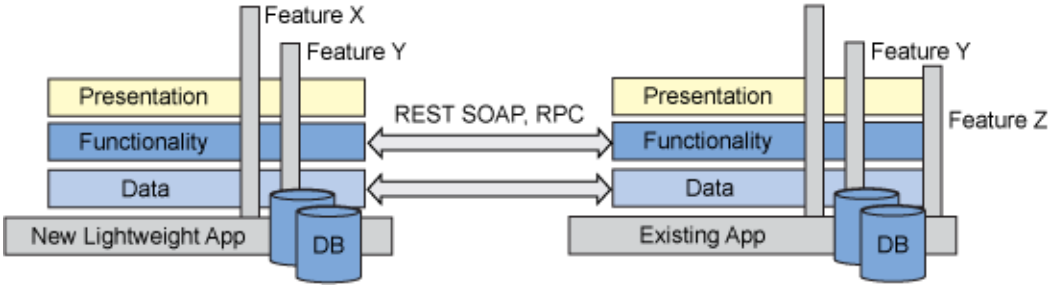
Figure 4. Option 2



Option 3: Create an API between applications, integrate data, and create new data

In this third option, you create an API between applications, integrate existing data, and create new data (see Figure 5). This style allows you to reuse functionality and data; it is strongly recommended that you choose this approach if you can't do the pure services model shown below.

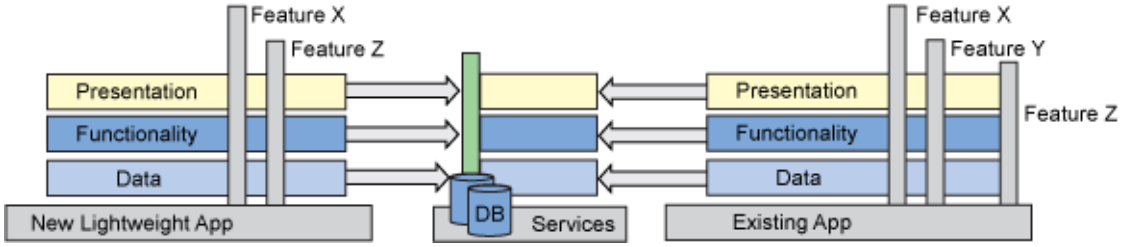
Figure 5. Option 3



Option 4: Create common services and have all applications use them

The optimal design is to create common services and have all applications use them. This option is the most difficult to do, but it is the most ideal approach. Always try this design and fall back to the third option if this option is not possible for you (see Figure 6).

Figure 6. Option 4



The types of lightweight applications requested by users

The most common types of lightweight applications that the business side of the hospital, or your provider organization will ask for will be in three categories:

- Additional functionality that allows MU requirements to be fulfilled.
- Mobile access to clinical data through new tablets, mobile phones, and other devices.
- Better communications between:
  - Providers and related providers (physicians to physicians or hospitals to hospitals)
  - Providers and their payers (providers send clinical documents and other materials to payers)
  - Patients and their providers (patients send information to their physicians)
  - Providers and their patients (providers send information to their patients without being contacted by patients)
  - Public health organizations and providers (organizations send notices and regulations to providers)
  - Public health organizations and patients (organizations send public health data to patients)
  - Payers and their patients (personal health records and other data transfers)

Users IT expectations don't fall into a neat single category. These expectations and the new regulatory requirements will require you to connect to your EHR system using one of the options mentioned in the previous section. This technical article does not get into the business details, but once you commit to create lightweight applications wrapped around existing or new services, you'll need to:

- Create a single-sign on and authentication mechanism so that your new applications don't require logging in each time.
- Create a clinical data repository (CDR) or central data mart that you can use for storing data from your EHR system (unless you plan on connecting directly).
- Connect to an enterprise service bus (ESB) that supports HL7 integration to populate your CDR or data mart.
- Store only small amounts of data in your new applications and retrieve other data from the CDR or central data mart.

- Build your applications using web technologies and HTML5 for mobile presentations.

Let's walk through each of the above tasks in the following sections.

User authentication, authorization, and single sign-on (SSO)

One attribute of lightweight applications is they proliferate. You start with one small one, then another, and then more. That's exactly what should happen but if you don't manage user authentication and authorization centrally and allow people to switch between these applications using a common login and password, you'll soon have applications that users do not want to use.

Luckily, there are many good options for common authentication and single sign-on (the common authorizations and role-based access control, or *RBAC*, is a bit more difficult). Let's get some definitions out of the way before looking at the options:

- **Single sign-on (SSO)** enables users to log in once to any application and then automatically be allowed to use any other application that uses the same SSO provider.
- **Common sign-on** enables users to log in to multiple applications using the same user name and password. This isn't as nice as SSO, but it is useful because it's easier to implement and, although it doesn't prevent multiple logins, it does not require separate user names and passwords for different applications.
- **Authentication**, provided by a user management system, confirms that a user is who they say they are. Authentication merely validates the identity of a person.
- **Authorization** is what a user management system does to verify what group a person belongs to, what roles they have in a system, and what specific permissions they should be granted in specific applications.

SSO Option 1: Atlassian Crowd

If you don't have a lot of authentication or SSO expertise in your organization, the Atlassian Crowd commercial SSO platform is a good place to start. It has many features and functions and can connect to multiple applications and directories via a single user name and password. Unlike many other options (like OpenID or CAS) it does a good job with group management and authorization in addition to the authentication.

SSO Option 2: CAS and SAML

If you have a good engineering staff, want to use open source products, and stay focused on open standards and protocols, choose common authentication service (CAS) for authentication and security assertion markup language (SAML) for authorization and user profile exchange. You can use CAS as a virtual login for

any back-end applications you have (after you write custom connectors or use the built-in ones for ActiveDirectory or LDAP). In order to pass user information from the common user profile, you use a SAML assertion to convey identity, groups, and related information.

SSO Option 3: Microsoft® Active Directory (AD)

If you're primarily working within a Microsoft Windows® environment you should try to tie all your applications together with Active Directory. Any .NET application and many PHP, Java™, and other applications can connect to AD (although .NET applications have the easiest time).

SSO Option 4: Lightweight Directory Access Protocol (LDAP)

LDAP out of the box can give you the ability to allow common sign-on as opposed to single sign-on. LDAP is already supported by most IT environments and is a good choice if none of the other previously mentioned options will work for you.

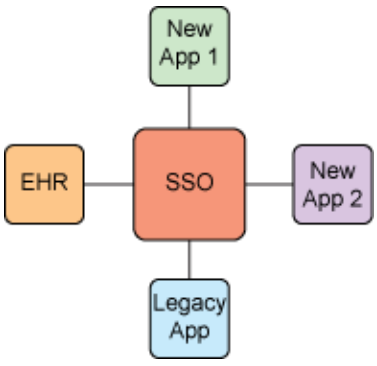
SSO Option 5: OpenID

OpenID is a common identity protocol supported by many web applications; it's not common to see OpenID supported by medical software vendors, but it is something you could build on top of your own internal products. Like CAS, OpenID is independent of the back end, language, and system. You can build it on top of almost any legacy application that gives you access to its user management system.

SSO implementation

Whichever SSO option you choose, be sure you have something in place that allows your new lightweight web applications and mobile applications to connect to their new functionality without having users learn a new username or password. If you don't provide common or single sign-on functionality, the chances for success of new applications will be hampered. As shown in Figure 7, the single sign-on solution should be a centralized service used by all of your applications, not a feature of any particular application. If SSO is properly deployed it will be easy to add functionality to your EHR because new functionality should decouple from the EHR, and pass through the SSO solution.

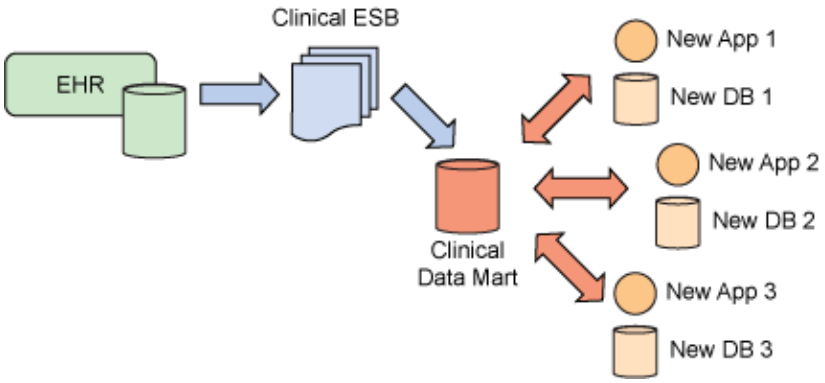
Figure 7. SSO choices



Building your new lightweight applications

If you're tasked with building new applications, your first requirement will be to do no harm, which means new functionality should never harm existing applications or data. Given that overriding goal, Figure 8 shows a good architecture to consider.

Figure 8. New EHR architecture



In this architecture, the new applications, which may be mobile, web, or desktop applications, should all have their own database for the types of data that don't need to be shared. Yet, they should communicate with a clinical data mart for all data that needs to be shared.

Create a common clinical data repository (CDR) or data mart

Working with your legacy or established IT systems might be difficult so you will want to consider a synchronized common clinical data mart where you can store EHR in a more accessible format. Consider your data mart carefully and only put the small subset of EHR data in it. Data marts are often created for analytical data, but given the power and performance of modern databases like IBM DB2®, CouchDB, and MySQL, the difference between analytical data marts and special purpose data stores like clinical data repositories is so small it is insignificant for the types of data you're considering here.

If you're not required to use a relational database due to corporate or other enterprise standards at your hospital, consider starting with a NoSQL database. CouchDB is a good choice.

Connect the CDR or data mart to an enterprise service bus (ESB) that supports HL7

The clinical data mart that shares data with your lightweight applications will need to be synchronized with EHR data. There are multiple options here, but you could start with enterprise-grade tools such as IBM WebSphere® Enterprise Service Bus, open source tools such as Mule or ServiceMix (and add HL7 parsing support), or choose a purpose-built HL7-enabled ESB such as Mirth. You might be interested in monitoring and putting the messages shown in Table 1 into a clinical data mart.

Table 1. Messages for a clinical data mart

Data Type	HL7 message type
Add patient	A04, A28
Update patient	A08, A31
Schedule appointment	S12
Cancel appointment	S15
Check-in appointment	A01, S13
Check-out appointment	A01, S13
Charge capture and billing	FT1 DFT^P03
Admit patient	Varies
Discharge patient	Varies
Update admission	Varies

Persistence for your lightweight applications

Although it is a good idea to centralize databases whenever possible, for lightweight applications that need to be created quickly and may be mobile, it makes sense to consider local databases for each of your applications or groups of applications. Keeping databases local and having a synchronization strategy gives your business users and developers a good deal of freedom and flexibility to create applications when needed and not accidentally harm existing legacy systems.

Instead of full relational data stores, you should consider NoSQL databases, such as CouchDB and MongoDB, for the local databases in your lightweight applications.

Web applications and HTML5 for mobile presentations

You'll want to choose standard Web applications or the new HTML 5 mobile standards as your presentation layer to keep applications as light and nimble as possible. Adobe Flash, Flex, AIR, Microsoft Silverlight, and Microsoft Windows desktop, and other options are favored as the business side asks for more interactive and visual functionality. However, by keeping your applications on HTML4 or HTML5, with some AJAX for interactivity, you will give your users the ability to run applications on iPads, smart phones, the upcoming Android tablets, and many other devices, including workstations.

Conclusion

In order to meet MU rules and provide the applications your users demand, you may need to build your own solutions on top of existing EHR systems. There are many choices you can make, but if you choose the service-oriented approach with a clinical data mart and HL7 ESB to separate the data, it will give you more flexibility and freedom to build new applications without being hampered by your legacy infrastructure.

Resources

Learn

- A Tour of the Healthcare New Media Marketing Landscape: Read a presentation about the various techniques hospitals can use to market their services online in this blog by Shahid N. Shah.
- HITECH Survival Guide: Learn about the Health Information Technology for Economic and Clinical Health Act (HITECH) Act
- Electronic Health Records and Meaningful Use: Review the Office of the National Coordinator for Health Information Technology's description of Electronic Health Records and meaningful use, and find links to information about the two incentive programs. One program encourages meaningful use by healthcare providers and the second program certifies EHR systems.
- The Nationwide Health Information Network (NHIN) is a set of standards, services and policies that enable secure health information exchange over the Internet.
- NHIN Direct: Learn more about this project to expand the standards and service definitions that, with a policy framework, constitute the NHIN.
- Open Health Tools: Learn more about this open source community that is dedicated to improving the health of people through the transformation of health information technologies.
- OpenExchange Project: Read how OpenExchange provides standards based core infrastructure to exchange patient health information in a secure and timely manner, to advance the quality, safety and efficiency of healthcare delivery.
- How Meaningful Use Impacts Healthcare Data Management Professionals: Learn what healthcare data management professionals need to know about achieving HITECH meaningful use and certification during this on-demand webinar.
- developerWorks SOA and Web services: Go here for articles, tutorials, standards, and other technical resources for Web services and SOA.
- developerWorks Information Management: Find IBM data management and database-related resources.
- IBM WebSphere Application Server: Build, deploy and manage robust, agile and reusable SOA business applications and services of all types while reducing application infrastructure costs with IBM WebSphere Application Server.
- developerWorks WebSphere Portal zone: Find information and resources on IBM WebSphere Portal, a composite application or business mashup framework and the advanced tooling needed to build flexible, SOA-based solutions.
- developerWorks podcasts: Listen to interesting interviews and discussions for software developers.
- developerWorks technical events and webcasts: Stay current with developerWorks technical events and webcasts.

Get products and technologies

- IBM product evaluation versions: Download or explore the online trials in the IBM SOA Sandbox and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- developerWorks blogs: Get involved in the developerWorks community.

About the author

Shahid N Shah

Shahid N. Shah is an internationally recognized and influential healthcare IT thought leader who is known as "The Healthcare IT Guy" across the Internet. He is a consultant to various federal agencies on IT matters and winner of Federal Computer Week's coveted "Fed 100" award given to IT experts that have made a big impact in the government. Shahid has developed multiple clinical solutions over his almost 20 year career. He helped design an electronic health record solution for the American Red Cross and deploy it across thousands of sites; he built two web-based EMRs used by hundreds of physicians; and he designed large groupware and collaboration sites used by thousands. As an ex-CTO for a billion-dollar division of CardinalHealth he helped design advanced clinical interfaces for medical devices and hospitals. Shahid also serves as a senior technology strategy advisor to NIH's SBIR/STTR program helping small businesses commercialize their healthcare applications.