Considerations in opening the mainframe to mobile devices

A guide for enterprise teams working on mobile applications

Leigh Williamson
Distinguished Engineer, Software Group
IBM

Skill Level: Introductory

Date: 05 Feb 2013

As more and more enterprises in all industries realize the need for mobile versions of their business applications, there is a need for an enterprise-class approach to mobile app development that leverages past investments in information technology and infrastructure. Leigh Williamson explains the five key themes of the IBM approach to mobile application development, which exploits existing information and transaction systems for maximum speed to market and reuse of services.

Overview

Across the globe, more people now use mobile devices as their primary means of obtaining information and requesting services over the Internet. This shift is due to the convenience of being able to carry devices wherever they go, along with their user-friendly and intuitive nature. Mobile application users expect access to accurate and up-to-date information, not time-delayed copies. This requires mobile applications to access and interact with the systems of record in the enterprise.

This crucial shift has motivated enterprises to develop mobile channels for their existing business applications and to plan for new kinds of applications that can exploit the unique characteristics of mobile devices. Enterprises have already made large investments in information technology systems and cannot afford to recreate these investments just to enable a new channel to their intended audience.

As with all major evolutions in the information technology industry, the first years of this shift have seen frantic activity to meet demand and create market presence. This initial frantic activity has been without considering more strategic issues, such as application development costs, maintainability, quality, and security. As the

© Copyright IBM Corporation 2013

mobile application market matures and the initial rush to market settles, those in the enterprise who are responsible for longer-term planning and economics are bringing these comprehensive software engineering issues into focus.

By leveraging the millions of lines of code in enterprise mainframe business applications and working processes, corporations can be even more competitive and strategic with the applications that they build. By moving from first-generation applications, through the Web, and now into increasingly sophisticated mobile channels, an organization should be able to deliver even more services to both internal and external customers. With the advent of Web and REST services, mainframe transaction managers such as CICS, IMS, and WebSphere Application Server can be excellent places to host services and processes that mobile front-ends invoke. Thus, the transaction managers can provide both information and business value. Often, these processes already exist but now need to be linked into the new open infrastructures that are available to the mobile platform.

IBM has established a reputation as a prudent and responsible software development partner for enterprises in many industries. This article focuses on a comprehensive approach for development of mobile business applications that leverage the significant investments in large transactional systems made by most Fortune 1000 companies. The technique described combines best practices for collaborative software lifecycle management with newer requirements that are unique to the creation of mobile applications. This approach is intended to provide value for all of the roles involved in mobile enterprise business application development projects:

- Architects who are planning for mobile projects
- Development teams that are making implementation decisions
- Project managers who are establishing the details of the project activities
- Test organizations that are addressing these new applications
- Executives who need an understanding of how these new mobile apps fit with the existing enterprise applications and development processes

Unique challenges for mobile application development

The creation of applications intended to run on newer mobile devices, such as smartphones and tablets, involves three categories of unique requirements and challenges:

- Form factors and usability
- Technology and implementation strategy
- Organizational implementation, application delivery, and collaboration

Form factors and usability

Several factors motivate the need for more attention to usability and user interaction design for mobile applications.

Form factors and user input technology

The first and most obvious unique aspect of mobile applications is that the form factor for display and user interaction is significantly different from prior forms of software. Smartphones usually provide only a four-inch area in which to display the application content. Even tablet devices have generally lower display sizes than PCs, especially when compared to the large flat-screen displays in use for newer desktop PCs. Significantly less data can be displayed at one time, so it must be exactly the "right" data, the data that is most relevant to what the user needs at that point in the application. User interfaces usually need larger widgets with more focused individual input forms than those that run on systems with larger screens.

Another obvious physical difference for mobile applications is that the mechanisms for user input are different. Mobile devices have pioneered the use of non-keyboard *gestures* as an effective and popular method of user input. Designers and developers must plan support for touch, swipe, and pinch gestures as part of a satisfying mobile application user experience. In addition to tactile user input, mobile devices are a natural target for voice-based user input. In fact, the traditional keyboard typing form of user input is probably the least effective and least popular mechanism for input delivery from mobile application users. Emerging use of stylus input mechanisms also offers new gestures to consider.

Unique forms of input offer new and valuable mechanisms to make mobile apps more powerful and useful than applications with a more limited array of input possibilities.

Besides input directly from the end user, mobile devices can receive input from other sources, such as geo-location input from the GPS component of the device, an accelerometer, and image information from the camera typically built into the device. These forms of input must be considered during mobile application design and development. They offer new and valuable mechanisms to make mobile apps more powerful and useful than applications with a more limited array of input possibilities.

Usability and user interaction design

Another factor is the difference in form factors and user input methods. It is much more difficult and time-consuming to plan how to display only the precise data that is necessary than it is to simply display all possible data, and let the end users visually sift through it for what they want.

The rich variety of input methods available on mobile devices is also a motivation for early design work. It is important to identify and use more efficient ways to deliver input data than the simple "just type it into a form" design that is a default for traditional web and PC applications. Extensive keyboard typing for mobile apps must be avoided to reduce end user frustration, particularly with drastically smaller touch keyboards and lack of traditional typing feedback. Identifying non-keyboard ways that

information can be gathered and delivered to the mobile app is a significant design challenge.

There is yet another more subtle motivation for extra attention to the design of mobile applications. The way in which end users interact with mobile devices and the applications running on them is different from how they interact with stationary PCs and even laptops. End users of a mobile device are typically holding the device in their hands while also interacting with their physical surroundings. These application users typically cannot concentrate intently on the mobile app for very long before they need to switch their attention to their surroundings. The interaction model for users of mobile apps is short, interrupted, and "bursty," meaning that they finish the application task very quickly before switching attention.

All of these factors drive the need for more investment in user-centered design for mobile applications very early in the development project. Ideally, these usability considerations and design aspects should be codified in the requirements for the mobile application. Then they can be linked to the later-stage development deliverables, along with the tests that validate that the usability of the app is as satisfying as possible.

Technology and implementation strategy

There is a spectrum of implementation choices for mobile applications. There is no one perfect answer for the choice of implementation for a mobile application, and all of the choices across the spectrum have their advantages and disadvantages. Thus, the challenge for mobile development teams is to understand the trade-offs between the technologies and make choices based on the specific application requirements.

The choice of implementation technology for a mobile project has an impact on other decisions related to the application's development. It might limit the choices for development tools. The implementation choice will probably have an impact on the team roles and structure. It might have an impact on how the application is tested and verified, as well as how it is distributed and delivered to the end user. Therefore, the choice of implementation approach for a mobile application is a crucial, early stage decision to make very carefully.

Native application implementation

A native implementation means that you are writing the application using the programming language and programmatic interfaces exposed by the mobile operating system of a specific type of device. For example, a native implementation for an iPhone will be written using the Objective-C language and the iOS operating system application programming interfaces (APIs) that Apple supplies and supports. Native application implementation has the advantage of offering the highest fidelity with the mobile device. Because the APIs used are at a low level and are dedicated to a specific device, the application can take full advantage of every feature and service exposed by that device.

Native implementations of mobile apps are not portable to any other mobile operating system. A native Apple iOS app must be totally rewritten if it is to run on an Android device with the two applications maintained in parallel. That makes this choice a very costly way of implementing a mobile business application.

Web applications

Newer smartphones and tablets come with advanced web browsers preinstalled. It is very feasible to implement a mobile business application that is a standard web application, plus special style sheets to accommodate the mobile form factor and to approximate the mobile device "look and feel." Mobile applications implemented using this approach support the widest variety of mobile devices, because web browser support for JavaScript and HTML5 is fairly consistent across devices. There are several commercial and open source libraries of Web 2.0 widgets, such as Dojo Mobile, that help with this approach. The web programming model for mobile application implementation also has an advantage for enterprises that already have developers trained in the languages and techniques for web application development.

One disadvantage of pure web application implementation is that such apps have no access to functions and features that run directly on the mobile device, such as the camera, contact list, accelerometer, and so forth. However, if your mobile application does not depend on local services running on the device, the pure web mobile application approach could be sufficient. As the HTML5 specification matures and becomes more widely supported by the mobile web browsers, many of the services local to the mobile devices will become exposed for pure web applications through that W3C programming standard.

A second disadvantage is that web applications aren't delivered through an app store. An app store has a presence, targeted entry point, and search mechanism. That search mechanism can be an advertising or access point for prospective customers that an organization wants to target.

Hybrid mobile application implementation

Hybrid mobile application implementation is a form of compromise between pure native implementation and pure web implementation. You write the mobile apps using industry-standard web programming languages and techniques, such as HTML5 and JavaScript. However, you package the app into a natively installable format that is distributed through an app store-style mechanism. This delivery model is well-understood and expected by mobile device users.

For the average mobile business application, many industry analysts have a strong conviction that the economics of code reuse and flexible application

development will favor the compromise hybrid approach over the long term.

A hybrid app is linked to additional native libraries that allow the app to have access to native device features from the single application code base. Because the bulk of a hybrid application is implemented by using technology not unique to any single device, most of the code for the application is portable and reusable across many different mobile operating systems. For the average mobile business application, many industry analysts have a strong conviction that the economics of code reuse and flexible application development will favor the compromise hybrid approach over the long term.

Organizational implementation, application delivery, and collaboration

Organizational change is often part of a company's move into mobile. Some organizations will create independent teams that manage the mobile applications. Other organizations will chose to create pockets of mobile focus in integrated business application teams. Either way, companies need to increase available skills and, at the same time, enable their processes through technology to support a much more visible process of application delivery.

Full lifecycle for the project

The lifecycle of a software development project generally follows a similar pattern, regardless of the type of software being created. It starts with the business decision, based on analysis, to invest in the application. Requirements for the application are captured and elaborated. These application requirements are further decomposed into user stories and feature work items, which are assembled into a plan of work for the iterations and releases to be completed for delivery of the application. Team members acting in various roles are assigned the work items and use various tools to complete the work and deliver that work to the project. The resulting application is tested and certified to deliver the requirements. The exact process and lifecycle followed for software projects at a particular company are usually tailored to that specific enterprise's goals and policies.

This lifecycle is the same for mobile applications, but more compressed. Mobile application development is generally characterized by small teams, by use of existing enterprise information infrastructure, and by highly user-interactive applications. Agile methods and test-first principles are ideally suited for such a scenario. Although the specific requirements for mobile app development are probably different from some other software development, the tools and processes for gathering, elaborating, and communicating those requirements are the same. In other words, the flow of the project and the need for integration and traceability across the project is the same for mobile and other software development projects.

It is also important to be able to link requirements and dependencies across multiple teams and applications in the enterprise. In this way, each team understands who they are dependent upon, as well as who is dependent upon them.

Integration of multiple tools

There are very few, if any, software development projects that can be delivered by using only one development tool. Most projects involve a wide range of tools from different vendors. Each meets the needs of a specific role or task in the overall lifecycle of the project.

For example, an individual developer of code for a mobile application might find that a simple code construction tool, matched to the mobile platform on which the application is to run, will suffice. However, that tool is missing features that facilitate the collaboration and coordination that are needed when an agile development team is involved in creating the application.

By integrating the individual developer's code construction tool with a compatible collaborative development platform that the whole team uses, agile teams can keep developers and testers on task and engaged. By doing that, they improve efficiency and quality.

Even though there are unique aspects to mobile application development, many of the roles and tasks involved in the development lifecycle are the same as for enterprise-class development of other kinds of software. In an article cited in the Resources section, "Measuring Agility and Architectural Integrity," Walker Royce describes the key techniques and practices for effectively delivering software by using agile and test-first principles. The software delivery practices in his paper are a perfect fit for mobile development projects.

Choosing a mobile application delivery approach

The IBM approach to mobile enterprise application development combines years of experience in the field of general enterprise software development processes with new tools and techniques that are specific to mobile devices and their underlying software.

Internally, IBM has gone through a transformation to employ agile methods across thousands of projects involving tens of thousands of developers working on teams that sometimes span multiple continents and time zones. With extensive expertise in the design and deployment of enterprise software across a wide array of industries, IBM offers customized solutions for the development needs of mobile application projects.

Collaboration across the extended team

Mobile applications are typically created by a small team with varying skills and expertise. A typical team might consist of a couple of developers of the fundamental

business logic and web services, a couple of user interface (UI) developers, a user experience designer, a few testers, and a team leader or manager.

Given the typically aggressive time frames for delivering mobile apps, even a small team must operate at peak efficiency. Any delay due to misunderstanding or miscommunication between team members can throw off the entire delivery schedule. Inception-to-delivery periods of a few months are common. The pressure to deliver mobile apps quickly results in the adoption of agile development methods for most mobile projects.

Mobile application projects supported across different mobile operating systems require code sharing and reuse. One developer might specialize in the code that runs on the mobile devices, and another might focus on the System z/OS code that supplies the critical business data to be shown on the mobile device. Clear understanding of the work that team members are expected to perform, and when they need to deliver it, is essential. Project requirements, timelines, plans, and so on are shared in that case, but source code, deployment platform, tests, and builds might differ.

Because of the strong business motivations to deliver mobile applications to the market quickly, mobile development projects typically have extremely aggressive timelines. Continuous integration and builds are important elements in agile development practices. Application changes delivered by developers for all of the mobile operating systems that the application is required to run on need to be processed immediately. If the mobile application is a hybrid or native implementation, several different builds of the application need to be triggered each time that developers deliver a change set for the application. The build setup and configuration for each supported mobile environment will be different from the others. It is likely that a small build server farm will need to be provisioned and available to handle these builds of the mobile application for multiple operating systems.

A broad development approach

As discussed previously, there are multiple pieces that make up the mobile application. You will need a development solution that addresses all of these pieces. Let's examine the various components of a potential hybrid mobile application based on the IBM Worklight mobile application development platform.

The first component, and the one that immediately springs to mind for most people when they think of mobile application development, is the user interface code and client logic that runs on the mobile device. The user interface code will consist largely of HTML, JavaScript, and CSS, with the client logic also implemented in JavaScript. IBM Worklight Studio provides a rich editing environment for this tier of the application. It supports traditional source code editing tools, as well as drag-and-drop visual design editing tools. Complementing this is a mobile preview capability that the developer can use to simulate the look and feel of different mobile devices.

It can also simulate capabilities of the native devices (for example, GPS, camera, and network status). This enables developers to perform significant validation of the mobile front-end before integrating their changes with the source code management system and sharing them with the rest of the team.

Next, consider the middle-tier systems. You might, for example, have one or more WSDL services running on WebSphere Application Server exposed through SOAP interfaces that will ultimately be used by the mobile client code. Here, you can leverage things such as Rational System Architect to design and model these services or Rational® Developer for zEnterprise® to develop and test the service implementations. Both products include IBM Worklight Studio in an Eclipse shell to simplify the overall environment for developers working on both aspects of the mobile application. Where appropriate, you can also combine those two applications.

Looking at your enterprise back-end, it is likely that there will be services running on your z/OS systems that you will need to use within the solution. For example, there might be a COBOL application made up of multiple CICS transactions that are exposed through RESTful interfaces. Here, you would look to Rational Developer for zEnterprise to develop the COBOL application or service processes, as well as the RESTful interface.

Now that it's understood that you'll need to use these middle-tier and back-end services from within your mobile client code, you might wonder how you actually make that happen without having to deal with the wide variety of enterprise service protocols. To answer that, turn your attention back to IBM Worklight.

The IBM Worklight server provides *adapters*, which are the bridge between the mobile client code and the various enterprise services that need to be invoked. They expose a common invocation interface to the mobile client code for all services, regardless of the service implementation and approach to exposing it. In addition, they handle such things as integrated authentication and security, as well as tracking for analytics. You can use Worklight Studio for both developing the mobile client code for development and testing of adapters.

Regardless of the actual development tools you use, be sure to remember that you'll need a development approach (and associated tools) that tackles all of the different components of a mobile application.

Testing across multiple and virtual environments

Another area where mobile application development poses a huge challenge is testing.

Testing for mobile applications represents a quantum leap in complexity and cost over more traditional applications. Unlike traditional PC and web applications, the range of potentially supported mobile devices and release levels is staggering. It is quite common to see test matrices for mobile projects that contain hundreds, even

thousands, of permutations of device, mobile OS level, network carrier, locale, and device-orientation combinations.

Testing for mobile applications represents a quantum leap in complexity and cost over more traditional applications.

The majority of mobile apps have a multi-tier architecture, with the code running on the device (the front-end client to data and services supplied by more traditional middle tier) and a data center back-end, where a services-oriented approach is used. These services might or might not be traditional web services. In many cases, the services are defined in more lightweight forms, such as RESTful service calls. Effective and comprehensive testing of mobile apps requires addressing all tiers of the application, not only the code running on the mobile device. The setup and availability of test versions of the middle tier and back-end services can present very large cost and complexity challenges for mobile applications testing.

Many mobile projects start by using manual testing approaches. This is the most obvious way to begin testing quickly. Manual testing is extremely expensive and inefficient, but it serves an important purpose by providing a mechanism for getting crucial usability feedback for the app.

You use mobile device simulators and emulators for some of what is known as *functional* testing. Using this approach, a software program running on a desktop workstation takes the place of a real mobile device. The use of simulators and emulators for mobile application testing can be valuable for tasks such as developer unit testing. Some of the device emulators are excellent, but some are not that good at replicating the real device. Therefore, in either manual or automated testing, some form of testing on the real mobile device is always essential.

In addition to testing the user interface or "client" side of the application, the services interface can be used as a way to speed up the implementation and testing of the "server" or *provider* side of these interfaces. The automated testing tools can act as a requester to the service, allowing the service provider to implement the service and test it without having to procure a myriad of different mobile devices.

Finally, there is a new class of automated testing tool that testers can use to create virtual service invocations. Virtual service invocations enable the services to be simulated, rather than running real-time tests. This increases testing flexibility and lets testing proceed both earlier and independently of development. This can result in greater assurance of the quality of the application.

A comprehensive testing approach

The implication of comprehensive, multi-tier mobile application testing is that more than one test execution capability must be used and coordinated into a single application quality result. IBM Rational Quality Manager software is an excellent

choice for tying together and managing the various test execution engines that are required for mobile testing. This test environment enables you to enact an integration testing-first approach that can help eliminate the big issues earlier in the lifecycle and improve economic governance. Walker Royce's paper mentioned earlier, "Measuring Agility and Architectural Integrity" (see Resources), describes in detail the technique for testing the hardest problems first.

There are many techniques in use in the market today for testing and validating mobile apps. Managers of an effective development project will employ all applicable techniques against a mobile application, because each technique has strengths and weaknesses. There is no best answer for mobile application testing, and the various techniques available are not mutually exclusive. The most effective testing strategy balances the use of all forms of mobile test and validation and compiles the results from each into a comprehensive mobile application quality metric.

Of particular value for enterprise mobile applications that employ services and data from IBM System z sources is the ability to isolate the expensive back-end System z interfaces and to simulate those when testing the code running on the mobile devices. (This is something that is likely to occur very frequently as the result of agile, continuous integration build output for the mobile app code.) This technique, known as *test virtualization*, is supported by the IBM Rational Test Workbench and the IBM Rational Test Virtualization Server products. Using this solution, test teams can avoid the need to set up complex middleware environments to support test execution for code running on the mobile devices. Rational Test Workbench can emulate the middle-tier and back-end services and protocols, so the test execution can concentrate on the client tier of the mobile app that is running on the device.

Conversely, the Test Workbench can also be used to simulate mobile application service requests. The enterprise application development teams can use this to test their changes and updates for implementing the service.

Integrated change management, software version control

Effective software version control is critical to mobile development. You need to be able to coordinate changes in the front-end code, the automated regression tests, and the infrastructure deployment scripts, along with changes in the middle tier and back-end services.

All code changes associated with a particular work item need to be tied together into a specific *change set*, or list of changed source code files. That set needs to delivered in one action so that the full code change can be tracked as a unit. Ideally, this process of assembling a change set should be as unobtrusive and seamless as possible, so that it does not interrupt the developers' concentration on the logic that they are creating.

The processes for version control and merge or rollback of code also need to be automatic and intuitive. Any time that a developer has to switch working context to

perform a task, it represents an interruption and a potential disruption or slow-down in the development process.

As mentioned earlier, most mobile development teams naturally gravitate toward agile development methodologies. Combine this with the need to coordinate with the teams that are developing the enterprise services, and it becomes crucial to follow agile best practices.

The usual agile team approach to software development is to define multiple short iterations in which they implement and validate a small set of application enhancements. A typical agile iteration is from two to four weeks long. The team leader can work with the team to map work items from a backlog list into the specific iterations and then assign corresponding work items to individual developers.

As the developers pick up the work items and begin to make progress on them, their efforts need to be automatically recorded and made available to the team leader to follow in a dashboard view. This makes the information about what is finished, what the team is working on at the moment, and what still needs to be done easy to track. Everyone on the team needs to be able to see how the iteration is progressing, along with the status of the work items planned for that iteration, without the overhead of having developers switch tasks to build status reports.

This kind of "whole project view" and end-to-end traceability is extremely important for any kind of software development project. It is especially relevant to mobile application development teams that are working on tight schedules and employing agile development methods. These kinds of lean development teams cannot afford to spend time tracking down details about whether and when a particular requirement was verified and delivered.

A collaborative lifecycle management approach

The IBM Rational solution for mobile business application development offers an integrated lifecycle management platform that supports collaborative tasks and helps link the various artifacts developed over the course of the product lifecycle. This solution also enacts delivery workflows and provides task management capabilities to run the mobile application development project effectively. It is augmented with integration of mobile-specific capabilities in the phases of the project lifecycle where such capabilities are needed.

Agile team collaborative development tools, such as IBM® Rational Team Concert™ software, enable the definition of multiple short iterations within the overall project. Work items can be moved easily from the project backlog to a particular iteration plan.

As the developers edit files in mobile application code-development tools that are integrated with the software version control system, a change set is automatically

updated and maintained. The developers don't have to do anything to produce the change set other than edit the files that they need to work on.

Change sets can be shared among members of the team before being fully integrated with the main code stream. Therefore, the web service developer who alters the format of data supplied by the web service can share a change set with the UI developer who is working on the logic that displays the new data, without the rest of the team being affected. When both UI code changes and web service code changes are consistent, compatible, and deemed ready, they can be integrated in one synchronized task into the mainline code stream for the rest of the team to pick up and use.

Summary

As more and more enterprises in all industries realize the need for mobile versions of their business applications, there is a need for an enterprise-class approach to mobile app development that leverages past investments in information technology and infrastructure. IBM has established such an approach that exploits existing information and transaction systems for maximum speed to market and reuse of services.

The IBM approach to mobile application development emphasizes five concepts:

- Simplify the mobile application user experience.
- Integrate first to improve economic governance of the mobile app project.
- Ensure traceability of requirements to tests that verify those requirements, by using work items associated with change sets during code development..
- Enact ultra-agile methods, tailored to the organization, for mobile development teams and enterprise application development teams.
- Evolve automated regression test suites for rapid deployment and testing to reduce the cost of change and to speed up feedback on the accuracy of changes made.

This approach enables mobile-specific tools and technology to be used with the same efficiency and rigor as other kinds of enterprise application development tools.

Read the appendix: IBM System z case study

Resources

Learn

- Learn more about the IBM approach to developing applications for mobile devices:
  - Explore the IBM Mobile Enterprise web page
  - Browse the IBM mobile application development page and the Mobile development section on developerWorks.
  - Watch the IBM Rational mobile overview video, narrated by Leigh Williamson, author of this article.
  - Listen to Rational podcast 242 (8:07): Announcing the IBM Mobile Development Lifecycle Solution: Design, develop, test and deliver high-quality, cross-platform mobile enterprise applications. There's also a transcript in PDF format and a white paper that you can download (requires registration).
  - Listen to Rational podcast 229 (8:07): Opening the mainframe to mobile devices. There's also a transcript in PDF format and a white paper that you can download, and a short video to watch (requires registration).
  - Modernize, extend and reuse The Jazz Revolution interactive (requires registration).
- Read: Royce, Walker. "Measuring Agility and Architectural Integrity." *International Journal of Software and Informatica*, Volume 5, Number 3, ISCAS, 2011.
- Find out more about IBM System z development:
  - Overview, features and benefits, and services web page
  - Solution brief: IBM Integrated Solution for System z Development: Accelerating multiplatform development (PDF)
  - Demo: Overview of IBM Integrated Solution for System z Development
  - Rational podcast 186 (14:03): Accelerate System z software delivery with IBM's Integrated Solution for System z Development (requires registration).
  - Watch the webcast, case study: Is there an app for that? Accelerate delivery and reduce costs with the IBM Integrated Solution for System z Development — one customer's journey
- Explore the Rational software area on developerWorks for technical resources, best practices, and information about Rational collaborative and integrated solutions for software and systems delivery.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools, as well as IT industry trends.
- Watch developerWorks on-demand demos, ranging from product installation and setup demos for beginners to advanced functionality for experienced developers.

Get products and technologies

- Download Rational Developer for System z for evaluation.
- Download a free trial version of other Rational software.

Discuss

- Join the Rational community to share your Rational software expertise and get connected with your peers.
- Rate or review Rational software. It's quick and easy.

About the author

Leigh Williamson

Leigh Williamson is an IBM Distinguished Engineer who has been working in the Austin, Texas lab since 1988, contributing to IBM’s major software projects, including OS/2, DB2®, AIX®, and WebSphere Application Server. He is currently a member of the IBM Rational software Chief Technology Office team, influencing the strategic direction for products in the Rational brand and leading the solution definition for mobile application development. Leigh holds a B.S. degree in computer science from Nova University and a master's degree in computer engineering from the University of Texas.

© Copyright IBM Corporation 2013
(www.ibm.com/legal/copytrade.shtml)
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)