WCTME Technology Roadmap

Technical Information Paper – v1.0

31 October, 2004

This documentation is proprietary to IBM and may be distributed UNMODIFIED to third parties without the prior written permission of IBM. Any unauthorized use or modification of this document is strictly prohibited. IBM owns all right, title, and interest in and to this documentation. Copyright © 2004, IBM All rights reserved.

Authors:

Name	Contact eMail
David Reich	wctmepvc@us.ibm.com

Attention: This document was printed from an online system and may be used only for reference purposes. The online document must be considered as the current valid version. Please ensure that this printed document is current, and to preserve its integrity, do not remove any pages.

Table of contents

1 Introduction

Workplace Client Technology Micro Edition (WCTME) is an integrated platform for the extension of existing enterprise applications to server-managed client devices such as desktop computers, laptop systems, personal digital assistants, (PDAs), and other mobile and pervasive devices. The integrated package combines the tools (WebSphere® Studio Device Developer [WSDD] and Micro Environment Toolkit for WebSphere Studio), run times (WebSphere Everyplace Micro Environment [WEME], Service Management Framework [SMF], and WebSphere Everyplace Custom Environment [WECE]), and middleware (DB2e, MQe, Web Services) for building, testing, and deploying server-managed client software to pervasive devices.

Over the years, computing models have moved from the server-based (mainframe) model to the original networked model (PC-LAN) with shared files, back to a server-centric (distributed application) PC-based model to reduce cost-of-ownership, back to a desktop, or device-centric model. It seems to cycle every few years. The only thing consistent in all of this is change, and the appearance is that the industry has not been able to find a good medium to maintain the autonomy and performance of localized applications with the cost benefits that come with server-managed environments. IBM® Workplace Client Technology (WCT) is the best of both.

In essence, WCTME is the foundation platform for the WCT family of offerings from IBM and provides a robust platform that supports devices from cell phones, PDAs and other devices to tablet, laptop and even desktop systems. Regardless of whether the computer is always, occasionally, or seldom connected, the WCTME model enables you to extend your applications and programming models using industry standards, without rewriting, to all of these devices.

The WCTME platform is built as a combination of a rich client platform (RCP – see Terminology in Section 2 below) based on the Eclipse model (originally devised for tools, and moved to the more generic application platform) as well as the browser-based model. Eclipse is an award-winning, open source platform for the construction of powerful software development tools and rich desktop applications.

The underlying premises of the WCTME platform are:

1. In this context, a device is not just a gizmo like a PDA or cell phone. A device is anything from one of these lower-powered devices up to a tablet, laptop or desktop PC.
2. The application model does not change from what is currently used, but rather is modified to make the assumption that the device is not necessarily always connected. While this is certainly the case for small devices, and even laptops, it could also be the case for a desktop. The model is to take what goes on today in the enterprises with Java™ 2 Enterprise Edition (J2EE) and Web container or rich UI presentation, and bring that to the lower-end devices without rewriting or re-architecting the applications.

To accomplish this, some application processing is moved from the server to the client. This is done for the times when the device is disconnected, so the user can execute transactions or just view data while the device is offline, and also, in some cases, just for enhanced performance. The application, depending on how it is structured, can process locally or against the server, or some combination thereof, depending on the programmer decisions and the state of connectivity at the moment the user is executing transactions. What makes this different from traditional distributed applications is that the underlying platform facilitates the deployment and maintenance of the code on the devices, and that connectivity and transaction processing can be managed dynamically without user intervention.

WCTME enables programmers to partition applications to run sometimes on the server, sometimes on the device, and sometimes on both, with the choice of presentation (rich UI or browser [Web Container]), data synchronization, application deployment and maintenance (with no user action required). In addition to the run times that make all of this magic possible is an integrated set of tools to make the architecture and creation tasks easier, and leave the developers the time to write efficient logic and snappy user interfaces.

Not only does the WCTME architecture provide for the scaling of applications and movement of parts to the server or client, but it also uses and open standard framework (OSGi$^{(TM)}$ /SMF) for deployment/management of these application components, or bundles.

WCTME is the next phase in application technology for lower total cost of ownership, while providing users with the flexibility to operate in any state of connectivity, and programmers with an extended J2EE application model down to the device, using standards-based technology. This can be done with little or no impact on existing application code or infrastructure, but rather with just a modification, or an evolution to the next level of user interaction.

This paper will give you the technological understanding as to how this is possible, and how you can take advantage of it in the programs you write or devices you build.

Unlike other papers you might have read, this is not a marketing paper. It will not describe the TCO benefits or marketing advantages of WCTME. Rather, it will tell you what you need to understand the technology and develop solutions using it.

2 Terminology

- **Workplace Client Technology Micro Edition (WCTME)**

  Provides an integrated platform for extending existing enterprise applications to server-managed client devices such as desktop computers, laptop systems, personal digital assistants (PDAs), and other mobile and pervasive devices. The integrated package combines the tools (WebSphere Studio Device Developer and Micro Environment Toolkit for WebSphere Studio), run times (WebSphere Everyplace Micro Environment, Service Management Framework, and WebSphere Everyplace Custom Environment), and middleware (DB2e, MQe, Web Services) for building, testing, and deploying server-managed client software to pervasive devices.

- **Workplace Client Technology Micro Edition Enterprise Offering (WCTME EO)**

  Provides an application platform for both end users and ISVs on Linux® and Windows® desktops, laptops, and tablets. The Enterprise Offering is designed for end users, system administrators, and ISVs. End users install the WCTME Enterprise Offering platform and then install and launch applications on the platform. System administrators install, configure, and manage the platform and applications. ISVs develop applications that run on the WCTME EO platform. The Enterprise Offering extends the functionality available in WCTME 5.7.1 to enable development of rich user interface applications that run on the Eclipse RCP.

- **WebSphere Everyplace Micro Environment (WEME)**

  IBM implementation of the J2ME specification that complies with the various licensing and functional guidelines to obtain the Java(TM) and Java Powered Logos.

- **WebSphere Everyplace Custom Environment (WECE)**

  Similar to WEME, but provides a more custom set of libraries and functions to enable customers and partners to create more tailored versions of the Java run times that are specific to their platforms, without necessarily needing to comply with established standards to get the Java logos.

- **J9**

  IBM's independent implementation of the Java Virtual Machine. The combination of J9 with the configurations and profiles create the run times. The configurations and profiles are Java class libraries.

- **Standard Widget Toolkit (SWT)**

  Analogous to AWT or Swing, this is a set of controls, panels, and other widgets that is essentially the building block of a user interface. The SWT provides a rich set of controls and is open source, along with the rest of Eclipse. Just like the RCP, an embedded version of the SWT (eSWT) is being developed for smaller devices. More information about SWT can be found at http://www.eclipse.org.

- **Eclipse**

  An open-source development framework (workbench) and set of widgets (SWT) designed for tools developers to leverage code reuse, a consistent user interface, and a plugin architecture for developing new packages. More information about Eclipse can be found at http://www.eclipse.org.

- **Rich Client Platform (RCP)**

  If you removed the tooling and IDE-specific functions from the Eclipse platform, what remains is the RCP. This RCP provides the core plugin environment along with a set of functions such as update manager, help subsystem and so on, into which application programmers can plug in their own code to create an integrated application execution environment. An embedded version of the RCP (eRCP) is being developed for smaller devices. More information about RCP can be found at http://www.eclipse.org.

- **WebSphere Studio Device Developer (WSDD)**

  Award-winning IDE from IBM for J2ME that extends Eclipse for the development of Java and C/C++ applications that run on pervasive devices, and forms the core of the WCTME tool chain.

- **Java 2 Micro Edition (J2ME)**

  Java platform for consumer and embedded devices. Like the enterprise (J2EE) and desktop (J2SE) platforms, J2ME is a set of standard Java APIs that delivers the power and benefits of Java technology tailored for consumer and embedded devices including a flexible user interface, robust security model, broad range of built-in network protocols, and support for networked and disconnected applications.

  - **J2ME Configurations** – Virtual machine and a minimal set of class libraries that provide the base functionality for a particular range of devices that share similar characteristics, such as network connectivity and memory footprint. Currently, there are two J2ME configurations: the Connected Limited Device Configuration (CLDC), and the Connected Device Configuration (CDC).
  - **J2ME Profiles** – Higher-level APIs that, combined with device configurations, provide a complete run-time environment targeted at specific device categories. Profiles further define the application life cycle model, the user interface, and access to device-specific properties.
  - **Mobile Information Device Profile (MIDP)** – Offers the core application functionality user interface, network connectivity, local data storage, and application management required by mobile phones and entry-level PDAs. Combined with CLDC, MIDP provides a complete Java Runtime Environment.
  - **Foundation Profile (FP)** – CDC profiles are layered so that profiles can be added as needed to provide application functionality for different types of devices. The FP is the lowest level profile for CDC. It provides a network-capable implementation of CDC that can be used for deeply embedded implementations

without a user interface. It can also be combined with Personal Basis Profile and Personal Profile for devices that require a graphical user interface (GUI).

- **Personal Profile (PP)** – A CDC profile aimed at devices that require full GUI or Internet applet support. It includes the full Java Abstract Window Toolkit (AWT) libraries and offers Web fidelity, easily running Web-based applets designed for use in a desktop environment. PP and PBP are layered on top of CDC and FP.

- **Service Management Framework (SMF)**

  An implementation of the OSGi Service Platform specification (SMF is currently at OSGi release 3). The Framework acts as a layer that enables operators to deploy multiple applications on a single Java Virtual Machine (JVM). Application developers partition applications into services and other resources. Services and resources are packaged into bundles, which are files that serve as the delivery unit for applications. Bundles have manifests with special headers that enable the sharing of classes and services at the package level.

3 Technology overview

It's important to take a moment to go over a little of the history that got things to where they are. In the embedded Java (J2ME) space, there are a series of configurations and profiles (see Terminology) that define which sets and subsets of Java class libraries (along with other new classes and methods specific for the embedded world) should be available to different classes of devices. There is no set specification that says that a cell phone with *x* amount of memory should be a MIDP device, or that a PDA should run only CDC applications, but they loosely map to classes of devices with different user interfaces, processor, and memory capacities. Along with this decision came the interesting question of how to get the applications onto these devices.

The OSGi Alliance (http://www.osgi.org) has defined a standard for management and deployment of applications. IBM has created an implementation of this standard called Service Management Framework (SMF). SMF is a set of run times that enables applications to be server-managed, and deployed to the client devices (regardless of device size, as long as it can handle the SMF platform code and CDC Foundation) without any specific user or administrator action. OSGi provides the architecture to enable JVM re-use and applications (and application components) to be dynamically deployed to a device, and started and stopped without tearing down and restarting the JVM.

SMF, and its tooling componentry enable a developer to break applications into manageable, deployable, scalable components, called bundles. Combined with other run times such as DB2e, MQe, synchronization technologies and others, SMF enables you to strategically break down applications into these bundles, allowing them to scale such that in a disconnected state, application users can query inventories, process orders or forms, queue transactions, update databases, and work in an offline state as though they were connected, with the exact same user

interface. This is done by keeping the programming model the same, breaking the applications into bundles and deploying them with SMF to cell phones, desktops, and everything in between.

What gets deployed where is a function of how the application is designed, its function and structure, and the Device Management Server, which implements the definitions and designs when the application architects tell it how the application bundles are to be deployed to which classes of device.

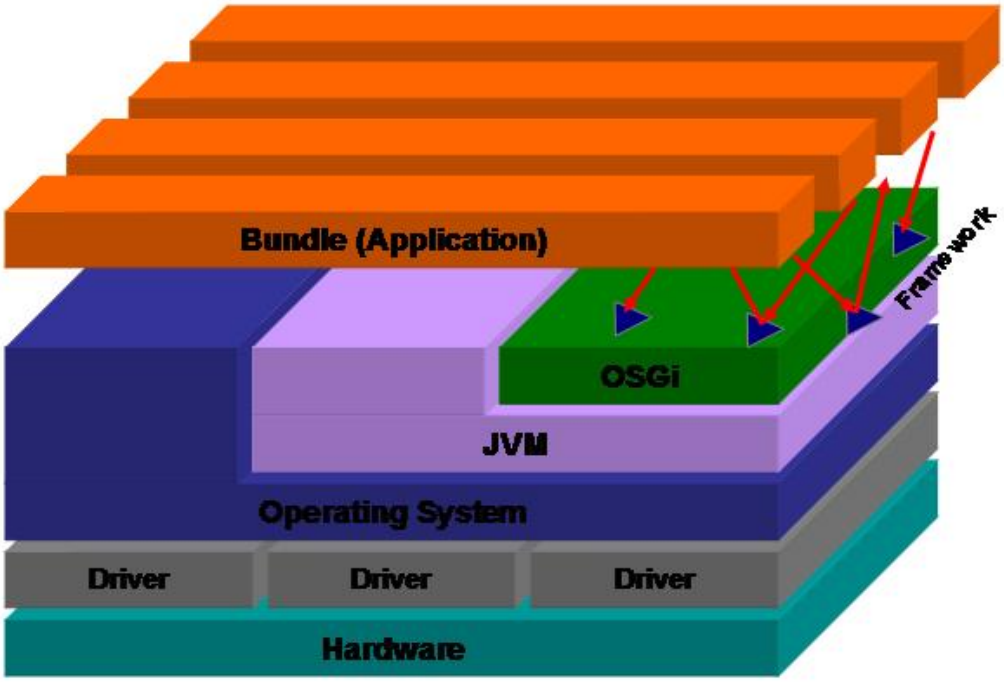

Figure 1. Platform structure

Figure 1 shows the generic structure of the run-time stack. You can see the underlying operating system platform, whether it be Windows, Linux, PalmOS™ or any cell phone or other device operating system. On top of that runs the JVM and the SMF platform. This enables the application loading and execution environment. On top of that is the variety of services the application might need such as DB2e, MQe and other application services or components. These can be written as high-level services, or even delivered as bundles themselves. This is the basic execution environment, and it is on this that the overall application presentation and execution is

based. In the next section you will see how all of the platform components, services, and applications fit together to form the user environment.

4 How the parts fit together

When an application is developed for WCTME, it is a Java application, and currently, this application is a J2ME application. (You will see in Section 5 how the WCT products extend to the desktop as well as the smaller devices). Figure 1 shows how the stack fits together. At the lower levels are the core of any computing device: the hardware, device drivers, and core operating system. On top of that sits the Java Virtual Machine (JVM). This JVM is really a combination of the JVM and run times for the device, where *run times* refers to the J2ME configuration/profile combination appropriate for that class of device.

This is a big decision point in the creation of applications. The reason is that SMF must have a minimum of CDC Personal Profile to run. SMF does not run on MIDP devices. As long as CDC/PP is there, the SMF model can be used. For lower end MIDP devices, more traditional application deployment and management methods must be used.

In the MIDP case, the application has to be deployed to the device through some mechanism other than SMF. The CLDC/MIDP configuration/profile is not sufficient to run SMF, so some other solution is required to get the application to the device. It can be preloaded on the device, either by a manufacturer, platform maker, or customizer (such as if devices are being deployed by an IT shop). Another alternative is to have the application deployed over the air by a service provider, or even installed by the user with some type of synch operation.

All of these methods work just as well for non-SMF CDC applications. However, with the SMF platform, applications (or more specifically, application bundles) can be deployed as needed, updated whenever there is a new version on the server, and dynamically started and stopped with multiple applications sharing the same JVM instance. While SMF does take up some resource, it provides a level of scalability and flexibility not before seen, even on many desktop platforms.

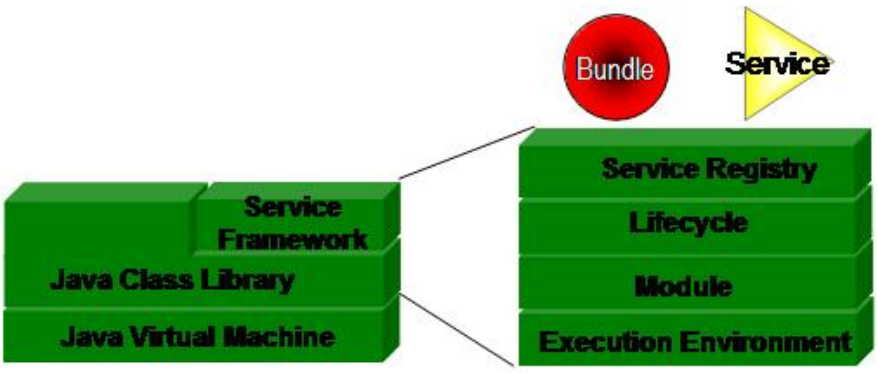

Figure 2. Service management platform

Figure 2 shows the service management platform and the where it fits on the Java stack. When a user of an application attempts to start an application, the appropriate bundle manifests are examined (Figure 3 below) for appropriate prerequisites or dependencies, and where needed, the device management server is consulted for what bundles are present. Without any user intervention, the appropriate bundles are delivered to the device so the application can be executed.

It is this SMF platform that enables bundle activation and lifecycle management of the application components, or bundles. As you will see in section 5, the tools provided for application development assist in the creation of bundles, with all of the requisite housekeeping such as creating BundleActivators and so on.

The other part of the picture, which is implicit in this scenario, is connectivity. Connectivity can take the form of wired LAN, WiFi, cellular modem, or any of a number of connection methods. The important thing to note here is that from the platform's perspective, the connection method does not matter. First, it is a layer below the platform. Second, the entire premise of these technologies is that they work connected, disconnected, or anywhere in between.

This discussion also provides the opportunity to mention WebSphere Everyplace Connection Manager (WECM). WECM is a connection stack that enables a user to not only tunnel into an internal company network securely, but enables a user to roam between networks seamlessly. An example is that of a user replicating a large database, connected with WECM. If the connection drops for any reason (driving through a tunnel, the WiFi link went down, and so on), WECM suspends the connection; when it is reestablished, the replication continues as if the connection had been alive all the time. This process does not involve the application. What this means is that the applications are truly separated from worrying about the connection state.

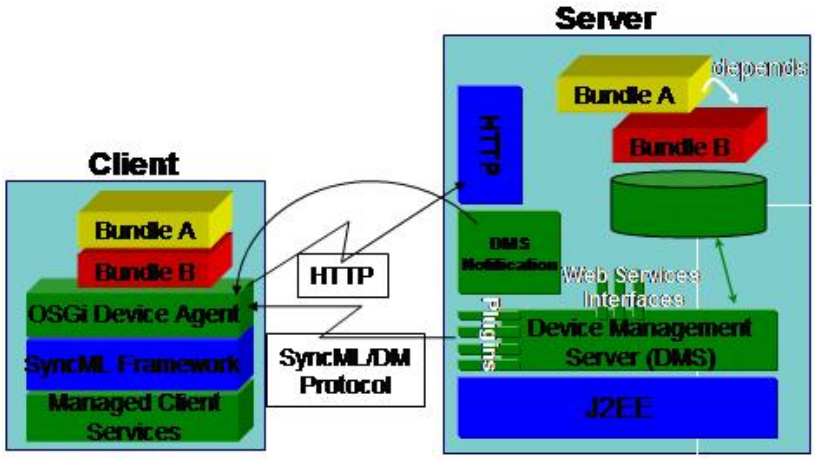

Figure 3. WCTME SMF bundle deployment

5 WCTME or WCTRE?

A robust set of program development tools is available to facilitate the creation of not only J2ME applications, but also Web applications and even J2SE applications, and structure them either as fully local applications, server-based applications, or anything in between. One question that has been asked recently has to do with the current IBM announcements of WCTME and WCTRE (Rich Edition). What's the difference? Where does each apply?

Both WCTME and WCTRE use the same Java and OSGi/SMF bundle mechanism for applications. Both offer programmers the ability to write bundles that extend the services provided. The easiest way to think about this is that WCTME is the core platform, whether J2ME-based or J2SE-based, with OSGi, and extension services including DB2e, MQe and others. WCTRE is a product offering that is a set of application components that performs a variety of office tasks such as e-mail, calendaring, word processing, spreadsheet processing, and so on. These components are extensible and customizable through a set of programming tools, but the primary focus of WCTRE is on these application components.

WCTME is the platform upon which WCTRE (and other OSGi/SMF, as well as other core J2ME applications) is built. Figure 1 showed the structure of that platform. If you plan to build applications to the base platform, you use WCTME. If you want to extend the office components, you use WCTRE.

In WCTME 5.7.1, which was released in August 2004, the package is built with the run times and tools for core J2ME applications. There is a WCTME 5.8 Enterprise Offering (EO) currently being built for J2SE applications that brings the OSGi platform to the desktop. In both cases, the user interface can be either a Web container (browser)-based UI, or a rich UI, written to the Eclipse RCP and SWT. In 2005, platform offerings will be available that combine this into one offering. The technologies currently exist, but they are segmented into J2ME and J2SE. In 2005, you will see the convergence of these segments, along with the tools and run times to support it end-to-end. There will be applications for Web container presentation, and also Eclipse RCP with SWT (eSWT and eRCP for embedded Java applications and devices).

6 Tooling

Through all of the work to build these offerings, IBM understands that without developers and tools, the run times are of little use. The core tool for WCTME applications is WebSphere Studio Device Developer (WSDD). This is a set of tools surrounding the core Eclipse platform designed primarily for J2ME application development. Figure 4 shows an overview of WSDD. WSDD is configured to fulfill the J2ME segment of application development, supporting the various configurations and profiles, and providing tools for profiling, optimization, debugging, and deployment to remote devices and emulation of devices, in addition to the standard Eclipse tools features and functions.

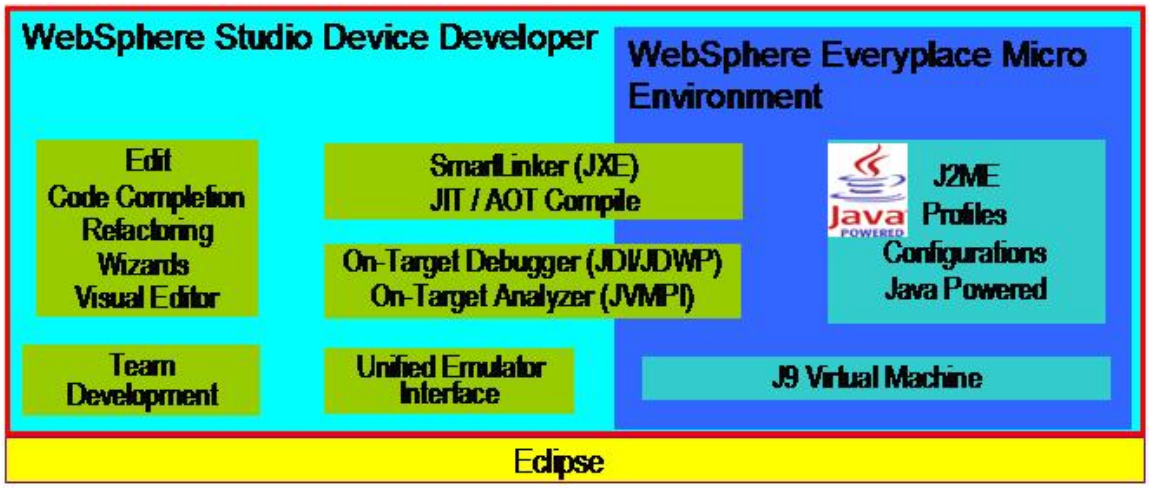

Figure 4. WebSphere Studio Device Developer overview

The OSGi/SMF tools such as the Bundle Developer tools are extensions to WSDD that are installable through the Update Manager function of WSDD/Eclipse. These tools build on the core J2ME tooling to enable enterprise functionality such as messaging (MQe), database (JDBC, DB2e, Cloudscape), Web services and of course, OSGi/SMF.

In addition to all of these tools in the WCTME package, WSDD has the ability to install, or link into a copy of WebSphere Studio Application Developer and WebSphere Studio Site Developer (WSAD/WSSD). This enables any programmer J2EE, J2SE, or J2ME (and even C/C++) to create an integrated application-development IDE with one user interface, one set of linkages to source code control and bug tracking system, one mechanism for team development and componentization, one set of debugging and help frameworks, and so on. In short, one tool to learn. Period. Not a tool for embedded, another for debugging, another for Web development, but one, componentized, customizable, scalable tool so people only ever have to learn one. What determines the combination of functions you use is your job function or programming task within the overall project. Referring to Figure 5, it is evident which pieces one would use, depending on the tasks to be completed.

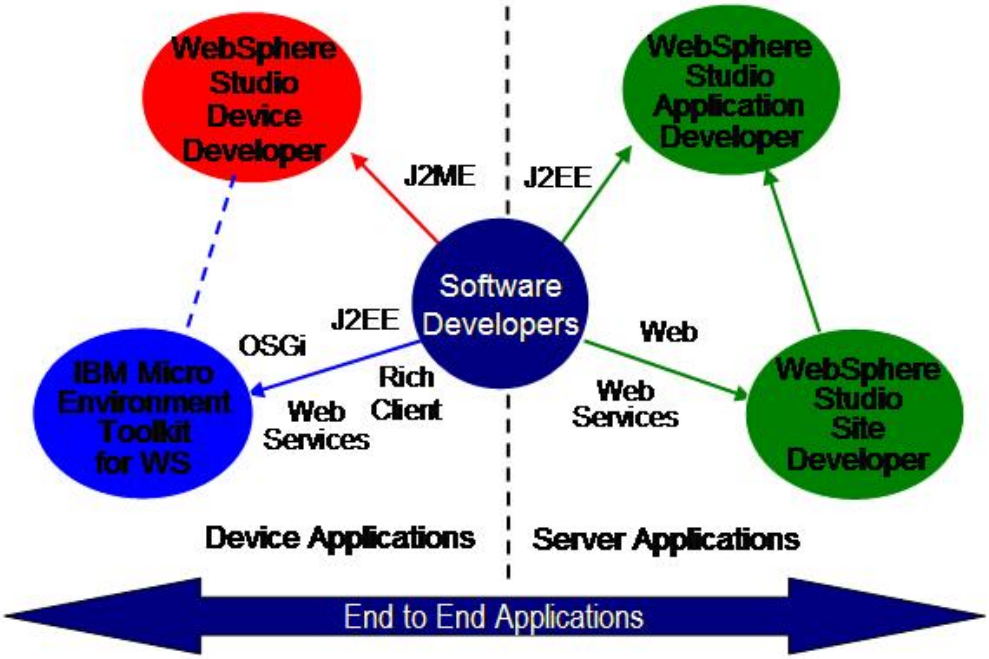

Figure 5. WCTME 5.7.1 tooling

As you can see in Figure 5, all of the tools interoperate and interlock. It's likely that no one person on a development team will need to perform all of the tasks across the tools, but what is important is that they share components so that tasks that are common across segments use the same tool plugins; developers can move among project segments and tasks without the learning curve of new tools.

The goal of tools in the past has been to make technology easier. A new approach has been taken in the tools with WCTME. Rather than trying to simplify the technology, these tools are designed around tasks, and how best to streamline a task. Ultimately, the goal is to have sets of application components that business people can assemble as full applications. To this end, the tools are currently geared towards creation of these components through wizards and builders, doing the bulk of the required work, enabling programmers to concentrate on the real logic of the programs. Ultimately, these components will be able to be assembled into applications.

Questions about tools in the past have been about which tools to get, and where. This is no longer the case. They are all packaged together, additional technologies are available through the update manager, and it all links into the enterprise WSAD/WSSD tooling.

7 What's Next

The IBM Pervasive Computing Division (PvC) is developing a comprehensive set of tools and programs that enable ISVs (whether they be application developers, enterprise developers, or platform or device manufacturers) to learn about and take advantage of this technology. Available in early 2005 will be a beta program of WCTME version 6 run times and tooling; in the meantime, you should keep on the lookout for a Redbook on WCTME (review comments welcome at http://www.redbooks.ibm.com). IBM currently offers workshops, and under development are technical papers, tutorials, and sample code for you and your customers and partners to try (check at the PvC Web pages listed in in Section 8). There are also newsgroups for WCTME and the associated toolset, WebSphere Studio Device Developer (WSDD) for questions, advices, and direct contact with the development and support teams. You can schedule briefings with the development team along with Business Development to pursue opportunities with your customers and ISVs. To schedule a briefing, send an e-mail to WCTME Partner/Austin/IBM@IBMUS or wctmepvc@us.ibm.com

8 Where to find more information

- Overview Whitepaper
  http://www.ibm.com/software/wireless/wctme
- WebSphere Studio Device Developer information and free trial
  http://www.ibm.com/software/wireless/wsdd
- IBM Micro Environment Toolkit for WebSphere Studio
  http://www.ibm.com/software/wireless/metws
- WebSphere Studio Device Developer Redbook:
  http://www.redbooks.ibm.com/redbooks/pdfs/sg247082.pdf
- WCTME 3-day workshop
  http://www.developer.ibm.com/events/wctme.html
- Workplace Client Technology Micro Edition Newsgroup
  news://news.software.ibm.com/com.ibm.wireless.wctme
- Free trial download of WCTME 5.7.1
  http://www-306.ibm.com/software/wireless/wctme/bundle.html

9 Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

IBM
WebSphere

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product and service names may be trademarks or service marks of others.

This product includes Eclipse technology.
(http://www.eclipse.org)

The End