Enabling Technologies for Branded Wireless Services

Boris Klots & Vadim Draluk
Motorola, Inc.
Challenges of Handset Software

- Increased demand for diverse functionality
- Handsets have specific limitations (memory, space etc)
- SW development cycle is long compared to marketing pressures
- How do we reconcile these demands and limitations?
- The process is similar to enterprise computing 12 years ago
- Let us look at these factors in more detail
Wireless Industry Trends

• Operator emphasis on branding
  – Branded UI & services

• Device Management
  – Decoupling of SW from life cycle of the device
  – Platform extension/upgrade
  – Software repair
  – Parameter tracking/configuration

• Proprietary OS ⇒ Open OS
  – Rising prominence of Linux
  – Continued growth of Java (MIDP2, JTWI)
  – Significant and expanding role for 3rd parties
Technology Requirements

- Deep UI and service postponement
- Richer services and applications to drive ARPU
- Portability and scalability
  - Across radio technologies (GSM, CDMA, 3G)
  - Across tiers
- Leverage industry technologies and trends
- Standards alignment, not fragmentation
Technology Requirements: Device Management

- Device Management that decouples life cycle of the device from the lifecycle of the software
  - Time to market
  - Innovation
  - Branded services
- Uniform Device Management that covers all stages of device lifecycle
  - factory, POS, customer care, consumer
- Extension of legacy provisioning/repair
  - Existing server infrastructure re-use
  - Sophisticated application provisioning with dependencies resolution
  - Software repair
- OTA capabilities easily available to applications
Key Industry Standards

- CE Linux Forum
- Java Community Process
- OSGi Alliance
  - Mobile Experts Group
- Open Mobile Alliance
  - WAP
  - SyncML
  - SyncML DM
  - OMA CP
  - OMA DRM
Software Platforms for Wireless Devices

- Modular framework for postponement
  - Drives operator branded services and UI
- Encapsulates generic dynamic (OTA, tethered) and security capabilities
- Allows to build end-to-end applications on top of readily available infrastructure
- Pervasive, standards-based device management
  - Enables superior customer care
- Signature user experience
- Encapsulates complexity of the underlying HW
- Reduced time to market
- Reduced COPQ
- Open standards-based
Software Platforms

- Drive cost, hardware portability, and improved time to market by using a standard operating system interface (POSIX) and open implementation (Linux)
- Drive reuse, security, and cycle time by leveraging Java
- Combine Java and Linux for best of both
Services

- Services packaged in modular units
  - Separately life-cycle managed (stop, start, …)
  - Separately provisioned, upgraded, and withdrawn
  - Abstracts the complexity of the service implementation from service consumers

- Services can be added, upgraded, and removed dynamically
  - The handset software is modular, extensible, and postponeable
  - Through a lookup mechanism, services can discover each other, and applications can discover services:
    - Services used by other services (SPIs)
    - Services used by applications (APIs)
Services Architecture provides flexibility to meet carrier-specific application requirements.
Service-Based Approach

Develop and test services, compose services into applications. Enhancements and bug fixes are implemented once for all applications. Application code is limited to unique flow and UI/navigation specifics.
Why A Java Based Approach?

The core platform functionality leverages the benefits of Java:

– Portability
– Integrity
– Security
– Large development community
– Familiar programming models
– Desktop development environments
– Standards compliance
Java applications layer on top and are isolated

Services and other applications are native and cannot be downloaded.
Expanding the Role of Java

- Java applications create signature user experiences
- Services in Java adapt framework to cellular
- Always-on JVM and application framework
- Open Operating System
- Abstracted modem interface

Modular, downloadable, field-upgradeable platform services and applications implemented in Java
What are the Implications of this Strategy?

- **Software Component Model**
  - Packaging
  - Provisioning and device management

- **Service Model**
  - Set of available services is dynamic
  - Varying degrees of trust between services

- **Applications Model**
  - JTWI and extended CDC application models
  - Common management of all applications

- **User Interface**
  - Rich, Java-defined toolkit
  - Common look and feel with integrated UI across all applications
  - Flexibility to support signature extensions
What is Device Management?

- Device Management Framework is the underpinning of the platform
- Provides a unified view on the data and makes OTA capabilities easily available to applications
- Device management comprises:
  - Content delivery
  - Content patching
  - Parameter tracking and configuration
    - making accessible a comprehensive subset of device state
- Device Management covers dynamic capabilities of platform from cradle to grave
- DMT: Common (among applications) data model providing OTA capability
Key Benefits of Device Management

**Operator**
Client-Server architecture enables rapid response to features and changing markets, consumer tastes, and branding

- Decouples service and device lifecycles
- Lower support cost
- Feature postponement
- Increased ARPU through rich messaging, more games, powerful new services

**Consumers**
Flexibility in choosing features and applications at the time of purchase and afterwards

- More compelling 3rd party applications and games
- Premium application upgrades available
- Updates to services for voice, messages, pictures, music and more
- UI personalization
Standards for Device Management

- Tracking and configuration of parameters
  - SyncML DM (aka OMA DM, new upcoming protocol for parameter configuration and manipulation of device management tree)
  - OMA CP (aka WAP Provisioning, older but still widely used protocol)
  - JUIX DM provides support for both protocols

- Provisioning of code
  - MIDP provisioning

- Software/Firmware OTA Update
  - Supported by newly introduced FOTA DM standard
  - Requires SyncML/DM and OMA Download capable servers
DM Services Architecture

DMT API
- DM Engine
  - Data Plug-in
  - EXEC Plug-in

Event Bus

Messaging

Repository

OSS Server

DM Server

Config Mngmnt Server

Tool
• DRM enables a content provider to attach rights to a piece of content
• OMA DRM v1.0 (Adopted Fall 2002)
  – supported by Motorola and other major handset vendors
  – defines right set and packaging
• OMA DRM v2.0
  – Stronger security (terminal keys, PKI infrastructure)
  – Specification due out in mid 2004
• Synchronization
  – SyncML DS
  – AirSync
Content Security: DRM

- app
- platform
- Linux OS
- content file
- rights file
Security

In addition to already known attacks, distributed service based architecture must be able to face traditional data network attacks

- Fine-grained access control allows to employ a more sophisticated security model based on degrees of trust

Mitigation:

- Secure Communication for client/ server interactions
- Java allows a fine grained access control on the device
- Extensive use of digital signatures to validate code and content
- Platform security services
- Hardware support for Security
Security: Chains of Trust

Verifying by Layered Chain of Trust

Certificate Chain

OTP

Hardware

SecureBoot
TCM

Trusted Root Public Key

Operating System (Linux)

OS Developer’s Public Key

Verifies then Executes

JVM + platform + Services/Apps

Platform Developer’s Public Key

Verifies then Executes

OS Developer’s Public Key

Platform Developer’s Public Key

Trusted Root Public Key

OSGi Alliance

MOTOROLA
intelligence everywhere

Copyright © 2008
Motorola, Inc.
Security: Multi-Layered Architecture
Conclusions

• The latest industry trends make service based architectures highly attractive.
• Multiple technical advances have made service based architectures not simply an advanced research topic but a business reality.
• Advances in hardware, maturity of Java make possible sophisticated client side frameworks.
• Donald Knuth: any CS problem can be resolved by additional level of indirection. The platform is a solution of this type.
• Maturity and acceptance of wireless standard protocols allow development of sophisticated end-to-end services.
• We discussed advantages and challenges provided by service and component based architectures.
• Opening up a device to data networks presumes a considerably increased security on the device.
THANK YOU