

An OSGi Profile for Embedded Devices

ESE 2009 - Ludwigsburg

André Bottaro, PhD - Orange Labs

Fred Rivard, PhD - IS2T

Wednesday, October 28th

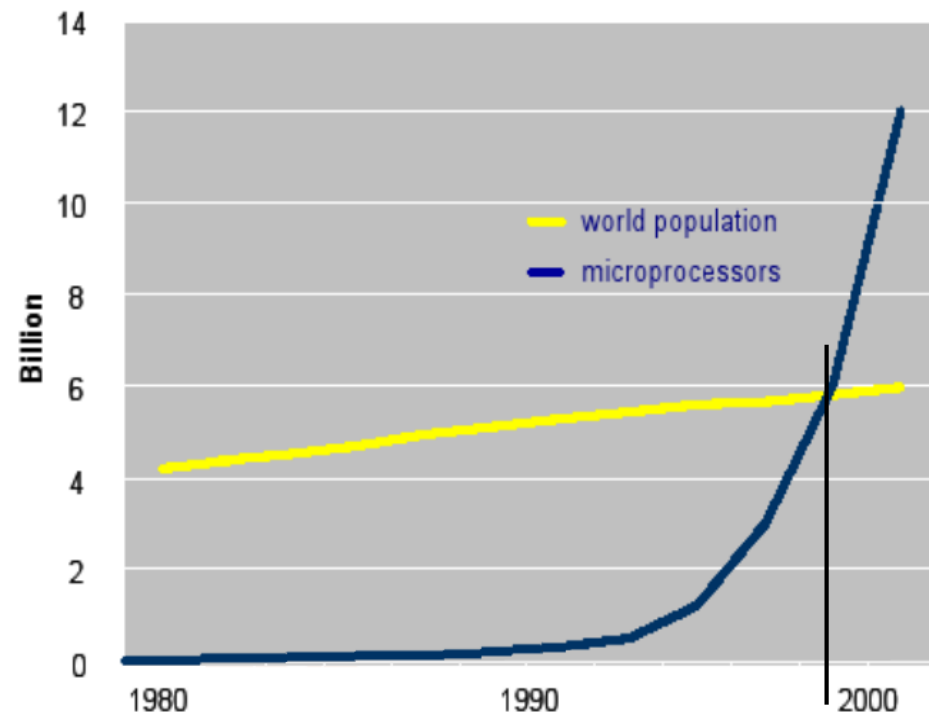
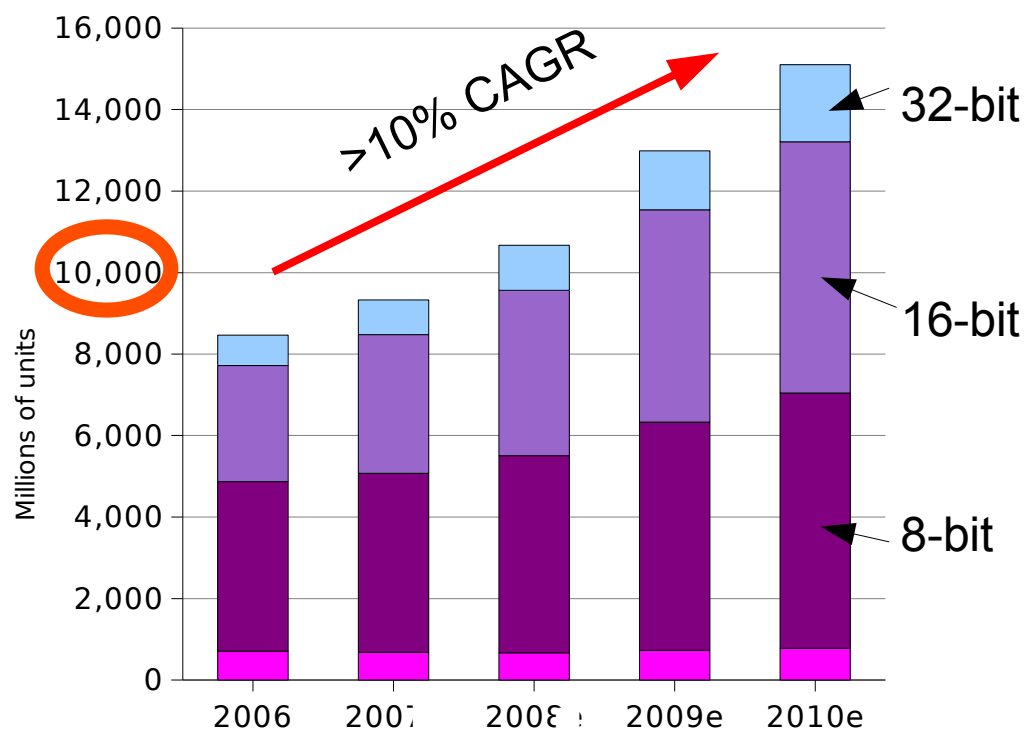
`andre.bottaro@orange-ftgroup.com / fred.rivard@is2t.com`

- **The Embedded world**
 - Market size, Markets trends
 - Embedded device == Economical device, Technical constraints
 - Life cycle of an embedded device, Bill-Of-Material
- **OSGi ME**
 - Why another profile for the OSGi platform?
 - Need to handle « cost » & « diversity » of embedded devices
- **OSGi ME in a nutshell**
 - Device Software Dynamics (DSD) & business cases
 - Fine-grained code sharing and isolation model
 - J2ME CLDC, OSGi binary compatibility
 - Same ordered initialization sequence (across several reboots)
 - Reliability : transaction, robustness, thread-safe

- Medical
- Security
- Telecoms
- Handsets
- Multimedia
- Home appliances
- Building automation
- Industrial control
- Automotive
- Transportation
- Defense
- Avionics



- The mcu market, the driving force is the 32-bit



2008	growth	
	units #	market \$
32-bit	+22.00%	+16.00%
16-bit	+11.00%	+2.00%
8-bit	+8.00%	-5.00%

2008	Market share \$	Unit price \$
32-bit	50%	5.00\$
16-bit	25%	1.40\$
8-bit	25%	1.00\$

2008	New products being designed
32-bit	59%
16-bit	20%
8-bit	14%

64-bit : 7%

Application specific for new projects				
%	2005	2006	2007	2008
Yes	60	57	57	48
No	40	43	43	52

**Agility
of development**



General purpose
Application Specific

Single core products

the commodity products

MCU

GPP

FPGA

DSP

MPU

Multi - core products

the innovative products

New Multi-Cores

FPGA High end

ASSP

ASIC / SoC

the products
developed
on demand



Performances

● current target
✦ futur target

Semiconductor
\$248B
iSupply2007

General Purpose IC's
\$15.5B

Dedicated IC's
\$81B

Micro controllers
Microprocessors
\$54B

Memory
\$58.5B

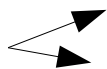
Discrete Components
\$16.6B

Optoelectronics
\$16.3B

- 98% of the processors are used in embedded devices

- Strong increase of software value in the added-value of a device
 - 2003 → 2009 : +87%
- Increased software R&D share in global R&D expenses
 - 2002: 31% → 2015: 41% (estimation)
- More software engineers

2 types of
software
engineers



Embedded Team Size	2007	2008	%
Software Engineers	6.3	8.1	+28.57%
Firmware Engineers	3.0	2.8	-6.67%
Hardware Engineers	4.3	4.3	0.00%
Total	13.6	15.2	+11.76%



- Project device life cycle : 2 phases
 - (1) Investment : 2/3 is for software
 - (2) Production : Bill Of Material per unit
 - The volume effect !
 - A small quantity is <75.000 units !

		benefit
Variable Cost		production & sales (IPs HW & SW)
		production & sales
Project Cost	Investments	Invest. HW /number of products
		Invest. SW / number of products

- **Reuse**

- Perpetuate SW and HW investments
- Obsolescence of HW and SW

2008

Project duration	13 months
Delay	4 months
SW Reuse	85%
HW Reuse	70%
Debug Cost	25%

- **Investments decisions based on financial indicators**

- Need of productivity
- Time-To-Market, ROI ==> Deliver On Time !

- **Software & Hardware heterogeneousness**

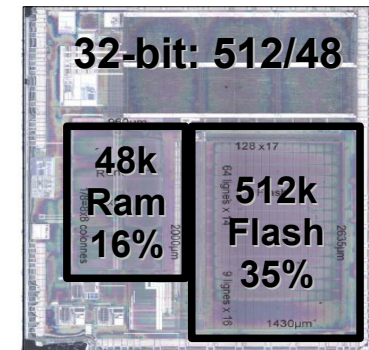
- High technical constraints : cpu budget, memory, consumption
- Reliability : certifications, safety critical

60% new device uses a 32-bit

70% of flash <= 512K

50% frequency <= 100Mhz

70% multi-tasks



The cost of a mcu is its physical footprint, often made of memories (ram & flash)

- A standard software environment ...
 - Software bricks from distinct service providers implementing common APIs
 - Benefit from the fast and abundant development of a wide community
- ... open to third party applications ...
 - Unleash creativity by assembling software bundles coming from third parties
 - Accelerate service delivery through an application store
- ... with agile development techniques.
 - Accelerate Time-to-Market with managed code development
 - Ubiquity (against hardware) thanks to code portability & reuse

« For [...] embedded devices, the main consideration is [...] no longer which OS to employ [...] but instead, which **application platform** to use [...] »

F.A.S.T, European Commission, Nov. 2005

- **The Embedded world**

- Market size, Markets trends
- Embedded device == Economical device, Technical constraints
- Life cycle of an embedded device, Bill-Of-Material

- **OSGi ME**

- Why another profile for the OSGi platform?
- Need to handle « cost » & « diversity » of embedded devices

- **OSGi ME in a nutshell**

- Device Software Dynamics (DSD) & business cases
- Fine-grained code sharing and isolation model
- J2ME CLDC, OSGi binary compatibility
- Same ordered initialization sequence (across several reboots)
- Reliability : transaction, robustness, thread-safe

- **Successful on the IT market**
 - Tremendous success on PC, thanks to Eclipse !
- **All OSGi fundamentals apply for the embedded market**
- **The market is demanding !**
 - A few success (vehicle, mobile, home) & lots of on-going R&D projects
- **But OSGi current version is “ too heavy ”**
 - Technical requirements, like for example class loaders, file systems, ...
 - Indirect requirement of MBs of resources & (very) high frequencies.
 - Incompatibility with the most widely spread environment: Java ME CLDC.
- **Need for a new profile to target Embedded Economical Constraints**



- **Availability of J2ME for typical 32-bit embedded MCU**
 - Typical embedded JVM footprint starts at 40KB (typical J2ME hardware starts with a 128KB of flash and 32KB of ram : ARM7, Cortex M3, AVR32, etc... with frequency ranging from 8 to 72Mhz)
 - Some JVM are baremetal with startup time below a few milliseconds
- **Driven by embedded markets business cases**
 - Keep OSGi high value features:
 - Fined-grained code sharing and isolation model
 - Dynamic software management (runtime, compiletime, linktime)
 - Software flexibility and Openness to Third Parties
 - Wide range of devices: from sensors to multi-services boxes
 - Only about Java (no native features handling in the specification)
 - CLDC Compliant & “Upward binary” compatibility of OSGi ME to OSGi
- **Orange Labs & IS2T : the RFP 126**
 - Orange & IS2T for the specification, IS2T to design one implementation.

- **The Embedded world**

- Market size, Markets trends
- Embedded device == Economical device, Technical constraints
- Life cycle of an embedded device, Bill-Of-Material

- **OSGi ME**

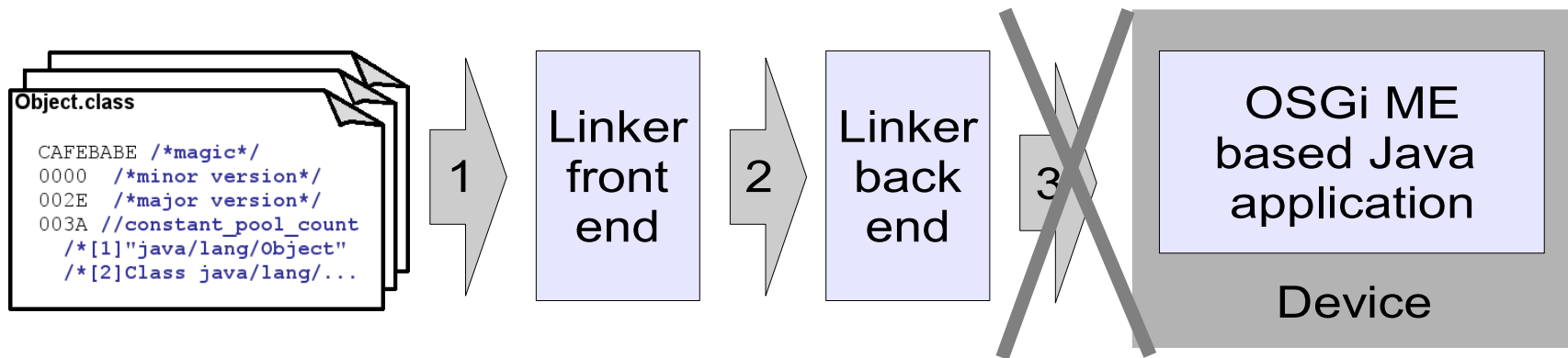
- Why another profile for the OSGi platform?
- Need to handle « cost » & « diversity » of embedded devices

- **OSGi ME in a nutshell**

- Device Software Dynamics (DSD) & business cases
- Fine-grained code sharing and isolation model
- J2ME CLDC, OSGi binary compatibility
- Same ordered initialization sequence (across several reboots)
- Reliability : transaction, robustness, thread-safe

- **Different market requirements on binary code downloads**
 - 1st case: no download due to certification, B.O.M., threats (anti-hack)
 - 2nd case: controlled downloads for only well-known (approved) services, threats controls, proprietary protocols via some typical media (uart, spi, i2c, CAN, usb, etc...)
 - 3rd case: Authorization of any download from any kind of sources
- **DSD levels characterize how binary code is loaded into devices**
 - DSD 0 = no download
 - DSD 1 = download through a controlled media
 - DSD 2 = no restriction

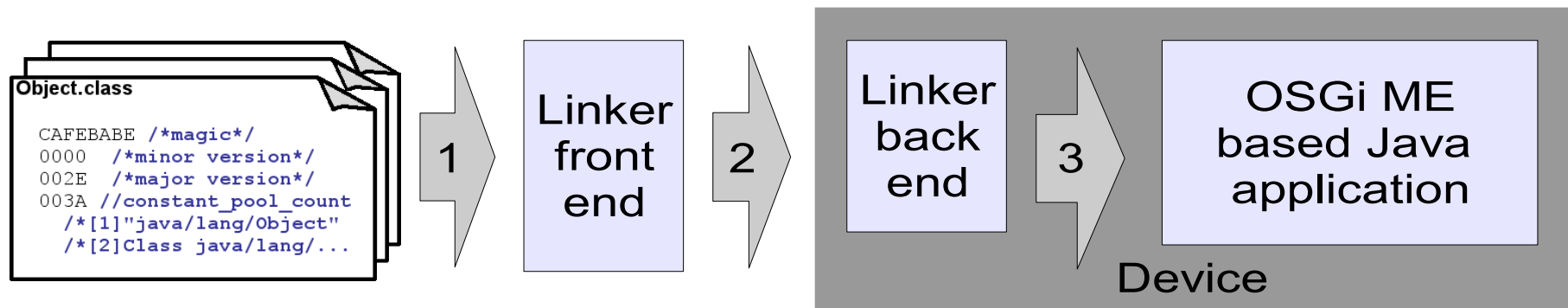
- The code is loaded using some probes (JTAG, ...) or bootloaders
 - The (Java) application is “switched off” to download the application
 - No runtime download



- A closed system
 - OSGi ME to design an component based application
 - The whole application is known at link-time (drivers, native codes, JVM, OSGi ME, CLDC and others libraries, the user Java application)
 - Linking off-board is feasible (only)

- **Code loading under strict control**

- The media by which the code gets downloaded to the device is controlled by some technical means, most probably protected by some proprietary protocol.

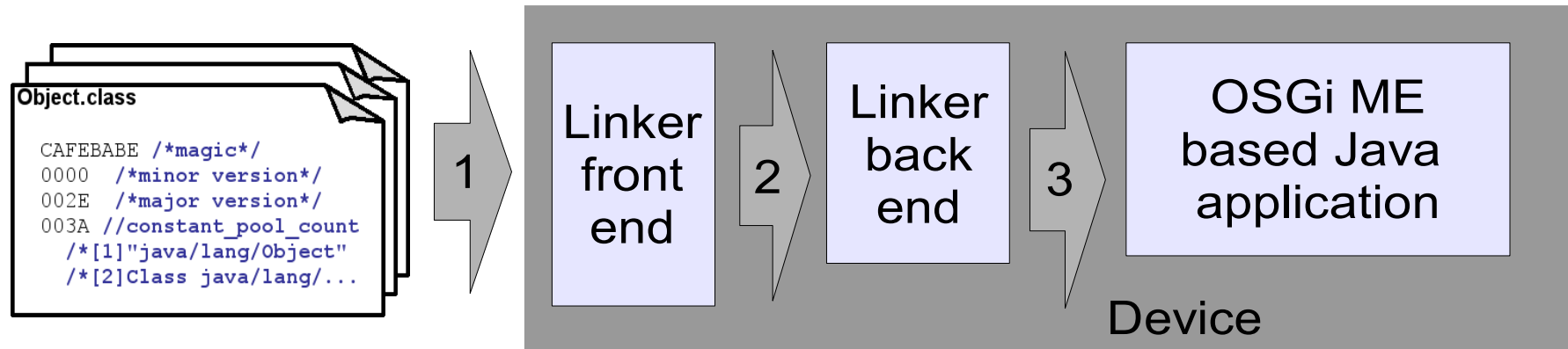


- **A controlled system**

- OSGi ME to design an component based application
- The application can download known bundles at runtime
- Pre-linking and off-board pre-analysis is feasible
- Adding and/or Updating are feasible

- **Free downloads**

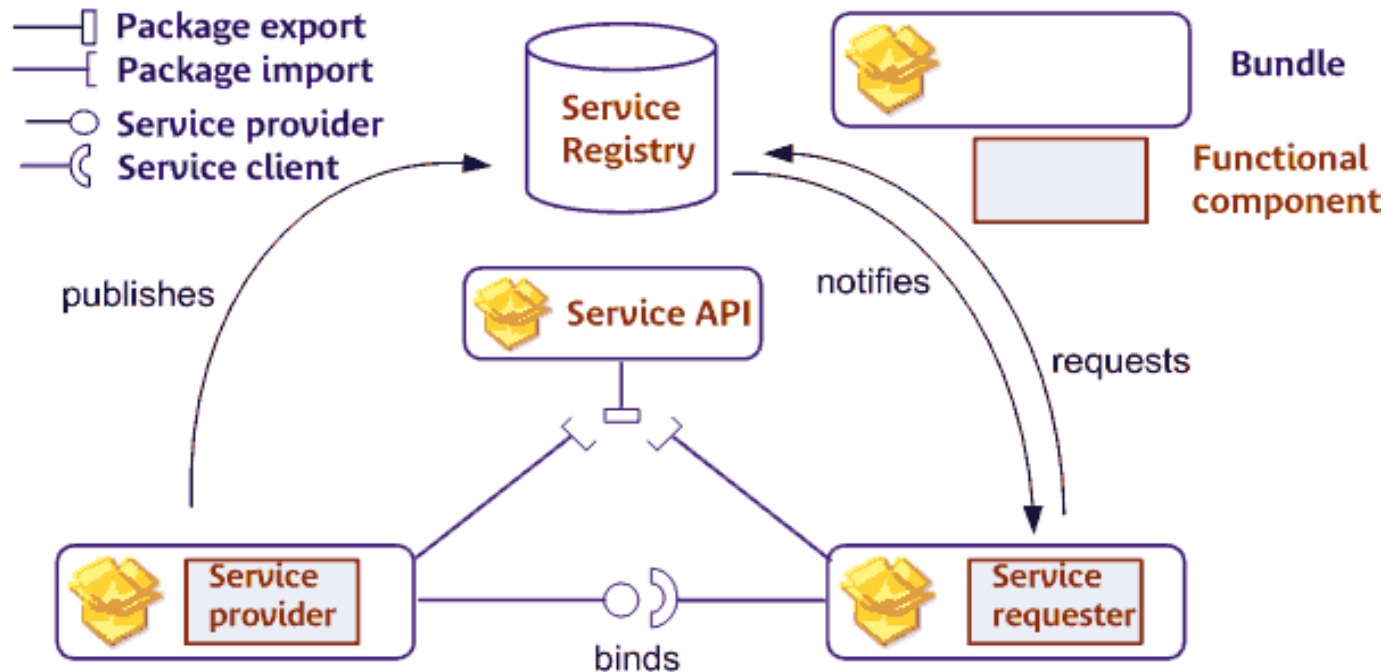
- The device has to embed all the necessary protections



- **A fully open system**

- OSGi ME to design an component based application
- The application can download known bundles at runtime
- All linking is done on-board
- Adding and/or Updating are feasible

- **Fine-grained code sharing and isolation between software bundles**
 - The foundation of the openness to third party application



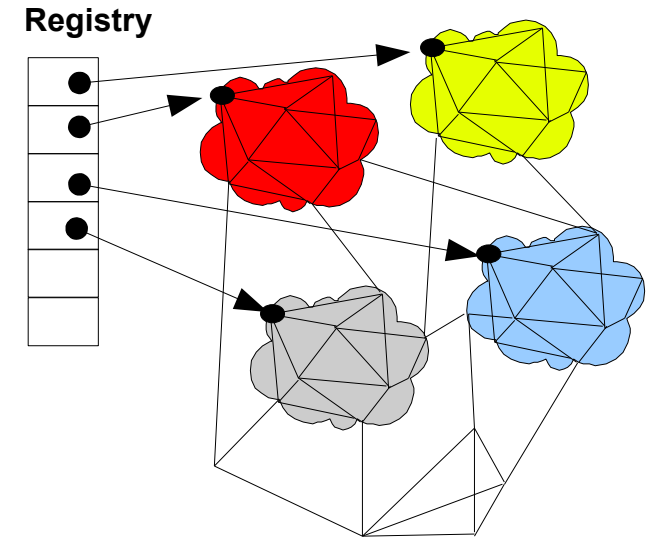
- **Service-Oriented programming relies on sharing and isolation**
 - Service clients and providers share public APIs
 - Service providers hide implementation details
 - Service clients use service providers through public APIs

- **The most wide spread Java runtime**
 - Thanks to the mobile phones industry
 - One good property: CLDC is small: it is ~50KB for the whole library
 - Small implies “less-code-to-execute” so turns to be fast.
- **Source compatibility**
 - Upward: An OSGi ME Java source is always compatible with OSGi
 - Downward: “Simply and automatic” renaming `Dictionary` to `Hashtable` will make the bundle “downward” compatible (if it makes sense !)
- **Binary compatibility**
 - Upward: the tool ME2O simply turns any OSGi ME binary bundle into a valid OSGi bundle
 - Downward: same kind of restrictions as for Source downward compatibility

- **Initialization of Java code: the intent**
 - *“The intent here is that a type have a set of initializers that put it in a consistent state and that this state be the first state that is observed by other classes.” (JVM specification section 2.17.4)*
- **OSGi ME re-enforces this intent: unambiguous start-up sequence**
 - No possible deadlock while the (Java) device starts.
 - Uniqueness of the “first-state” (same for each device's start).
- **When DSD > 0**
 - Compatibility of downloaded bundles arrival with the initialization semantic
 - Group-Of-Bundles: within a group, bundles are ordered according to the initialization sequence (whatever is that order until it is compatible with the Java semantic).
 - Persistent storage: ordered GOBs (Groups-Of-Bundles)

- **No stale reference**

- A faulty service cannot cause the failure of the entire platform.
- A service == a graph of cooperative objects, with a root that is registered into the Registry.
- Accessing a dead service throws a runtime exception
- No manual service reference counting.



- **Responsibility of using a service**

- Need to anticipate a possible dead service
- Provisions for such situation (same as exception handling)

- **Thread Safe framework**

- Any method can be called several times from several threads concurrently on the same receiver without jeopardizing the state of that receiver.
- The user application may be synchronized on any accessible object of the framework without causing a deadlock with the implementation of the underlying platform.

- **Small number of critical sections**

- This definition allows the number and the size of critical sections to be minimized, without compromising the robustness of the framework.
- The user is free to synchronize its application design according to its need without risking mysterious deadlocks.

- **Reliability**

- Each thread possesses a unique transaction it can enter in, several times (beginTransaction, endTransaction).
- While executing a transaction, all modifications on objects cannot be seen by other threads
- If an exception occurs while the transaction executes and is not caught inside the transaction, an exception is thrown at the beginning of the transaction (roll-back)

- **Memory side effects visible at the end of the transaction**

- Upon success, all modifications are made visible to other threads. All the assignments are made sequentially, at no particular order, and not necessarily atomically.
- Use of regular synchronized statements, when required by the application

- **OSGi ME framework uses all the robustness features for its needs**

- **Avoid complex features**
 - Hosting (exporting) several versions of a same class
- **Removal of features that are against a strict sharing and isolation declarative model**
 - dynamic imports
 - optional imports
 - bundle requirements
 - bundle fragments
 - bundle extensions

- **OSGi ME requirements**
 - Keep the core features of the OSGi technology,
 - Be compliant with Java ME CLDC,
 - Simplify OSGi technology,
 - Strengthen robustness.
- **Your feedback is welcome to reinforce the current proposal**
 - OSGi ME : the RFP126 from the OSGi Alliance

Thank you !

`www.is2t.com`
&
`www.orange-ftgroup.com`

? Questions ?

`fred.rivard@is2t.com & andre.bottaro@orange-ftgroup.com`