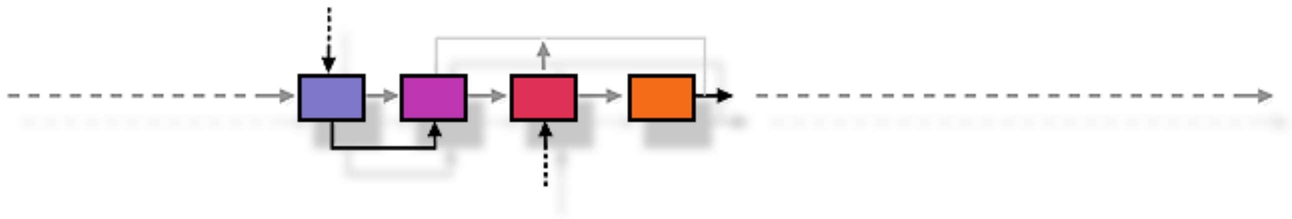


Advancing Java embedded

May 8 2017

Leonardo Lima,
Heather Vancura

Working group members

- Eclipse
- Fujitsu
- Gemalto
- London Java Community
- MicroDoc
- Oracle
- V2COM
- Werner Keil

Goals

- The JCP EC established the Java ME Working Group to address the concern of EC Members about the current state of Java ME
- Goals:

1. Describe technical and non-technical requirements for Java to be relevant and useful in embedded devices and IoT.
2. Provide grounds for discussion with Java leadership within Oracle on how to improve Java ME, or the next version of Java for Micro devices.

Use cases

- There are many different IoT use cases, with completely different set of technical and business specifications. We're focusing on two different device classes that group some of the desired characteristics
- **Specialized device** - targeted to a specific function, and it runs on specialized hardware. This means that its overall capabilities won't change over its lifetime.
- **Gateway device** - has a broader scope of function, when compared to a specialized device.

Technical Specifications

Performance

- Specialized devices are associated with tasks that require fast startup time
- Specialized devices are also sized to have just enough processing power, to save on costs
- Gateway devices normally don’t suffer performance penalties

Size

- There are two size constraints on specialized devices: RAM and ROM/Storage.
- RAM consumption on specialized devices is critical
- ROM (Flash or Disk storage) is also a critical constraint: storage memory on these devices is costly and in many processors, it’s built in into the processor
- For gateway devices, this is not always an issue
- For specialized devices - merge all binaries - VM and User Code, “gutting” the VM and taking out unused classes

Language compatibility

- Java ME 8/MEEP was a huge step towards SE compatibility and SE Embedded profiles made SE more like ME when comparing sizes.
  - But it’s either very small runtime without JNI or a bigger runtime without some of the ME frameworks

Language compatibility

(Still) Missing in Java ME:

- Collection streams
- Reflection
- Runtime Annotations
- Concurrency utilities
- Collections and Math APIs

Language Compatibility

Useful IoT frameworks on ME missing in SE:

- Software provisioning
- Software management
- Application concurrency (MVM)
- Inter-application communication (IMC)
- Events
- Service Provider/Consumer pattern

Native Code support

- One of the more concrete things missing for embedded development is better operating system/C/C++ integration.
- Libraries like JNA and Device I/O is a step in the right direction, but these aren't part of the Spec or the default platform.

IoT specific or new APIs/JSRs

- Emerging standards for IoT communication today are supported in Java (SE/EE) via JSRs (like JSON) or not (like MQTT and COAP).
- Suggested areas for standardization
  - REST client
  - Communication protocols such as MQTT and/or COAP
  - Device I/O (it does exist today, but it’s not a JSR)
  - LWM2M

Update of old JSRs

- JSR 177
  - Security is a critical aspect of IoT, and JSR 177 provides Java ME applications with APIs for security and trust services through the integration of a Security Element
- JSR 361
  - Needs to be realigned with state-of-the-art features and current embedded device market requirements.

Business Specifications

Licensing

- Current business model for Java ME does not scale.
- It still uses the same old licensing model, while Java SE moved on to a more open/public implementation that has a frictionless technology onramp where technologies can be implemented, tested and standardized.

Development Model

- Similar open collaborative development concept for Java ME as for Java SE (one example, For Java SE →OpenJDK + JSRs). Which implies that everybody can get the source and port it to a target system.
- For business reasons, key technologies still need to be defined as JSRs.

Board support packages

- Because embedded development is not a box package, there's the need for easier download availability from hardware manufacturers.
- Hence, there is a need for a binary to download with porting compatibility so that hardware vendors can easily port the standard implementation to their hardware/boards

Other Considerations

Broad set of requirements

- Every vertical and even use case in IoT has a different set of requirements
- The current way of sharing a single platform/binary for the whole JSR specification may be outdated.
- A better way would be a legal and safe method to pick-and-choose the functionalities/classes that we need for our target environment.

Low Java ME 8 adoption

- Business related aspects contributed to low Java ME 8 adoption immediately following the release.
- The technology met the needs, but the costs to migrate were too high.
- In the meantime, most of these issues have been resolved. So a next version, with updated technology and licensing terms might see (much) better adoption.

JSR Leadership

- How does Oracle want others to contribute? Does it want to lead or allow others to lead and collaborate in this space?
  - Some JSRs are either touching Oracle IP (such as Device I/O) or are already lead by Oracle (such as JSR 177), so we need a clear position from Oracle on such issues
- Most members on this Working Group would be willing to participate in JSR activity, but not necessarily lead.
  - Eclipse is interested as a potential source for communing members interested in leading JSRs
  - V2COM is interested in leading some of the JSR efforts
  - Gemalto is interested in leading JSRs in the Java ME domain

Market Demand

- Java is the only standardized platform which scales well. Many people are not aware of this.
- Java as an IoT platform also offers the developer appeal of an end of end technology platform.
- Is there (still) a market for Java ME on feature phones?

Java Community Process

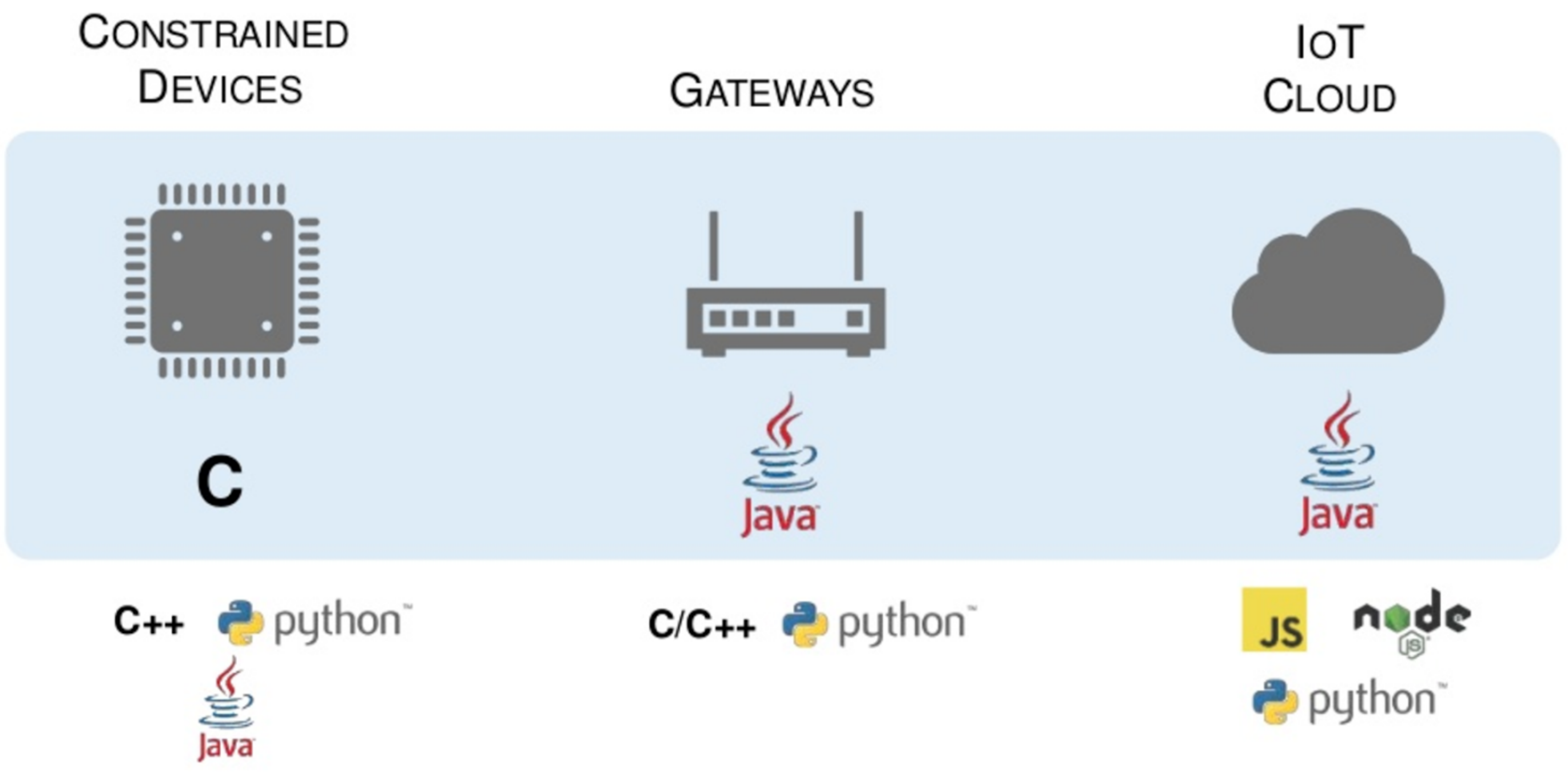
Competing technologies

- Node.JS (and embedded JavaScript)
- Android/iOS
- Android Things
- Windows IoT
- Ubuntu Core
- Eclipse Edje and “small Java VMs”
- mbedOS
- OSGi

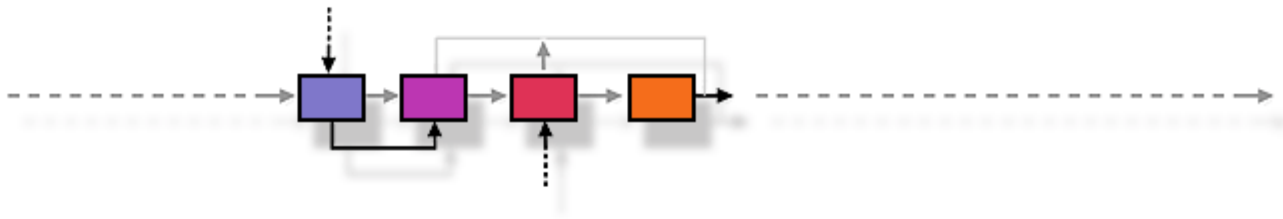
Eclipse IoT survey

- Eclipse IoT Developer Survey. You can find a written summary here and here, and the full slide deck here.
- The good news is that Java is doing very well on the gateway and the cloud. On the cloud, node.js and JavaScript are growing fast. In fact, if you add JS and node together, arguably JS is the dominant language.
- On IoT Cloud Platforms, Java (46.3%) emerges as the dominant language. JavaScript (33.6%), Node.js (26.3%) and Python (26.2%) have some usage.

Eclipse IoT Survey



Questions, discussion, next steps



Thank you!
http://jcp.org