Typescript support for Intel Edison
Why, how, Expected benefits

HEADS Project
Olivier Barais
INRIA
barais@irisa.fr

IOT Days, Grenoble, March 2015
HD-Services

• Heterogeneous and Distributed Services
  ▪ **Heterogeneous:** The infrastructure on which the service runs is composed of a set of different nodes and networks.
    - The "Future Computing Continuum" which ranges from microcontroller based sensors and devices to cloud.
  ▪ **Distributed:** The implementation of the services is composed of a set of independent processes communicating asynchronously.
    - Truly distributed services implementation is required in order to provide useful and reliable services which take advantage of the infrastructure.
Examples

- Health domain and ambient assisted living
- Energy domain and smart grids
- Environmental monitoring and oil and gas
- Safety in hazardous environments
- Intelligent Transport Systems (ITS)
- ...

IOT Days, Grenoble, March 2015

Why?
Why "HD-Services"?

- Isn't Internet of Things about having everything connected and available in the cloud?

End-users

Service developers and providers

Centralized implementation "in the cloud"

Service

Cloud(s)

IOT Days, Grenoble, March 2015
Limitations of centralized approaches

• Very easy to develop, evolve and maintain but...
  ▪ Underexploits "Things" capabilities
  ▪ Does not allow real-time or critical services
  ▪ Not resource efficient (bandwidth)
  ▪ Not robust
  ▪ Does not scale

Good solution when possible but not sufficient in many realistic cases
Distributing the implementation

- The service implementation is distributed to exploit the infrastructure

Why?
Benefits of HD-Services

- Complex to develop, lots of different skills involved but...
  - Allows fully exploiting the features of each platforms
  - Allow for local and/or decentralized decision making
  - Robust to partial and/or temporary failures
  - Push processing close to data sources
  - Allow for real-time and critical services
  - Can scale in a "big data" context

In practice for more and more real-world services are HD-Services

IOT Days, Grenoble, March 2015
What are the problems? (1/6)

- Here is an example infrastructure
What are the problems? (2/6)

- Here is the software components needed for the service
Why?

What are the problems? (3/6)

- Heterogeneous infrastructure and technologies are needed
What are the problems? (4/6)

- A lot of different expertise are needed
  - Both for development and runtime deployment/maintenance
What are the problems? (5/6)

- Someone needs to coordinate all experts
  - Design the different components, their functionality and interactions

Client-side Javascript

Proprietary Protocol

Database

Web Interface

Cloud back-end (eg. Linux/JAVA)

Gateway (Embedded Linux)

Sensor/Device (Microcontroller)

IOT Days, Grenoble, March 2013

http://heads-project.eu

Why?
What are the problems? (6/6)

- Large heterogeneous teams need to collaborate
  - A service architect / developer
  - Many "platform experts"
  - Complex and expensive
  - Unavailable to small actors
- Service maintenance and evolutions
- Infrastructure is dynamic
  - Constant evolution/adaptation
- (Early) Validation?
- Software reuse?

**Challenging and expensive**

IOT Days, Grenoble, March 2015
Challenges

• Provide a support for a multi-views point approach for IoT on top of Eclipse
  ▪ A kind of Eclipse Polarsys Capella for IoT ;)

• Viewpoints examples
  ▪ Manage development of large applications
    ▪ Typescript example
  ▪ Manage the deployment and evolution of large scale distributed applications
    ▪ Kevoree example (www.kevoree.org)
  ▪ Event streaming management
  ▪ Reactive programming (ThingML.org)
HEADS Goal

• Provide tools and methods
  ▪ For each actor to concentrate on his task
  ▪ For decoupling the tasks of different actors
  ▪ Using state of the art software engineering practices
    0 Modularity, reusability, runtime deployment, continuous integration, validation, etc...
  ▪ Cost efficient and practically usable
    0 No large overhead, integrated with legacy systems, etc...

Why?
Using state of the art software engineering practices

A support of typescript for Intel Edison

A viewpoint for service developer

IOT Days, Grenoble, March 2015
What is TypeScript?

- Free and open source, strongly supported by Microsoft
- Based on ecmaScript 4 + ecmaScript 6
- Created by the father of C# Anders Hejlsberg
- A superset of JavaScript
- To answer why we need JavaScript+, we need to understand what's wrong with vanilla JavaScript
What is the problem?

- Why do people hate working in JavaScript?

Using state of the art software engineering practices ;)

IOT Days, Grenoble, March 2015
What is the problem?

- JS is designed for small things
- We now use it to do big things
- But JavaScript is not suited for building large applications
- Your JavaScript code gets complex; it becomes extremely unwieldy
Let's look at TypeScript

- To get started with TypeScript, grab it from http://typescriptlang.org

- Let's look at TypeScript, first the core concept...
TypeScript - first glance - optional strong type checking

- // js
  function f(x, y) {
    return x * y;
  }

- // ts
  function f(x: number, y: number): number {
    return x * y;
  }

// Type information is enforced in design and compile time, but removed at runtime
TypeScript features

- **Static Type Checking**
- **Modules and Export**
- **Interface and Class** for traditional Object Oriented Programming

- Works with all your existing JavaScript libraries
- Low learning overhead compared to similar JavaScript systems (*CoffeeScript* or *Dart*)

- Amazing Visual Studio, eclipse or IntelliJ tooling
- Outstanding team and refactoring scenarios
Summary - why TypeScript

- Have to learn one more thing - there is a learning curve, very easy if you already know JavaScript, or if you know C# or Java very well.
- You still have to learn JavaScript - Understanding how TypeScript converts to JavaScript will teach you better JavaScript.
- Some definition files don't exist or incomplete, but I think this will vanish very quickly. These aren't hard to write if you really need something.

- **Modules** and **classes** enable large projects and code reuse
- **Compile-time** checking prevents errors
- Definition files for **common JavaScript libraries** makes them very easy to work with, and provides strong type checking
- Source map **debugging** makes debug easy

IOT Days, Grenoble, March 2015
How?
MRAA

• Libmraa is a C/C++ library with bindings to JavaScript & python to interface with the IO on Galileo, Edison & other platforms, with a structured and sane API where port names/numbering matches the board that you are on

• We need an interface definition for libmraa (mraa.d.ts)
  ▪ Generate mraa.d.ts from .h file
    https://github.com/HEADS-project/mraa_hpp2ts_generator
  ▪ Or get it from github
    git clone https://github.com/HEADS-project/mraa
Test it

- First install the node.d.ts
  > npm install tsd -g https://github.com/borisyankov/DefinitelyTyped

  and next you can download it in typing
  > tsd query node -a install #Download node.d.ts

  Use the following command to compile your typescript:
  > tsc --module commonjs AioA0.ts

  next you can install mraa in this folder:

  > npm install mraa

  finally you can run the samples e.g. AioA0:

  > sudo node AioA0.js

IOT Days, Grenoble, March 2015
Using MRAA definition for typescript

```typescript
///<reference path='typings/node/node.d.ts' />
///<reference path='../../src/typescript/mraa.d.ts' />

var m = require('mraa'); //require mraa
console.log('MRAA Version: ' + m.getVersion()); //write the mraa version to the console

var analogPin0 = new m.Aio(0); //setup access analog input pin 0
var analogValue = analogPin0.read(); //read the value of the analog pin
console.log(analogValue); //write the value of the analog pin to the console
```
More complex example

```javascript
import fs = require("fs")
import http = require("http")
import path = require("path")
import mraa = require("mraa")
import express = require("express")
import index = require("./routes/index")
import user = require("./routes/user")

var col = fs.readFileSync("collections/collections.js", "utf8");
eval(col);
```

IOT Days, Grenoble, March 2015
Initial conclusion - if I have to make a decision for you...

- If you see yourself using more JavaScript. You have to look at TypeScript.
- If you and your team has to work on JavaScript together, you have to look at TypeScript.
- Once you've done the initial hard work and converted a project. You can't stand going back.
Next steps

- Complex deployment of HD-Services
  - A configuration language for managing module deployment and reconfiguration
Kevoree (www.kevoree.org)

- Kevoree project aims at supporting dynamic adaptation in distributed system (What Benjamin calls Management layer)
  - MDE@Runtime
    - Shared model representation for distributed nodes
    - Offline & online operation, compute@Model level, apply @Runtime
  - Component-based
    - Actor semantics on each ports to closely separate component behavior
    - Communication semantics between component in channel
    - Support component reconfiguration (parametric, architectural, behavioral)
  - Continuous Design, type definition continuous definition
    - Hot (re-)deploy & provisioning
  - Heterogeneity management with Node Type
Kevoree general overview
An OSGi-like framework for HD-Services
Time for Demo
Thank you!

• Questions?