Construction d'applications dans la jungle de l'IOT

DIVERSE : DIVERsity-centric Software Engineering
Software intensive systems
Heterogeneous and Distributed Services

- **Heterogeneous**: The infrastructure on which the service runs is composed of a set of different nodes and networks.
  - The “IOT Jungle” 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)
Heterogeneous infrastructure
New Services on top of the IoT Jungle
<table>
<thead>
<tr>
<th>Arduino</th>
<th>Raspberry Pi</th>
<th>Intel Edison</th>
<th>ST Nucleo</th>
<th>ESP8266</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3-$30</td>
<td>~$35</td>
<td>$40</td>
<td>$8</td>
<td>~$3</td>
</tr>
</tbody>
</table>

**Programming?**

- **C/C++**
- **Python/Java/C++, Javascript,…**
- **C++, C++/Lua**

**Modelling & generating!**

- **USB/ SPI, In-Situ**
- **Serial/ OTA**
- **Built-In**

**Arduino Uno:**
- 13 GPIO/
- 1 UART
- 6 analog inputs.
- 1 UART (Rx/Tx).
- 12-bit ADC with 16 Timer (8)
- I2C (3), SPI (3)
- USART (3)

**Power consumption**

- 12V 50mA
- Deep sleep: 88mW
- Idle 88mW
- Powered off 45mW

**ESP8266:**
- 3.3V
- Transmit 802.11b, CCK 11Mbps, POUT= +18.5dBm 197mA
- Standby 0.9mA
- Deep sleep 10 µA

**Storage**

- Built-In
- Serial/SD Card

**I/O**

- Arduino Uno: 13 GPIO/
- 12-bit ADC with 16 Timers (8)
- I2C (3), SPI (3)
- USART (3)

**Code distribution**

- USB/SPI/Serial
- In-Situ
- Built-In
Model-driven software engineering to handle diversity
Our Main Objective

Enable efficient exploitation of the broad diversity of the future computing continuum for rapid service innovation of advanced services by providing service developers with new agile tool supported software engineering methods enabling short innovation cycles.
Heads project outputs

Two new modelling languages to tame the IoT jungle
What is ThingML?

A DSL to model distributed reactive systems
- IoT systems, embedded systems, sensor networks, ...

Components, State machines and action language
- « Main stream » MDE

Contributions of ThingML
- « Complete » action language
- Slots, Mixins and Aspects instead of Inheritance and Composites
- Enforced encapsulation and actors semantics

Target Platforms and Applications
- MDE for resource constrained systems (microcontrollers, IoT)
- Development of applications distributed across heterogeneous hardware
Why ThingML?

Typical MDE benefits
- Reduce development, maintenance and evolution costs
- Perform verifications and analysis on the models
- Model application at a platform independent level

No existing approach can deal with microcontrollers
- ThingML can run on hardware less than 1ko of RAM

No existing approach is really platform independent
- Since actions are written in the target language
thingML Goals

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
  • Modularity, reusability, runtime deployment, continuous integration, validation, etc…
• Cost efficient and practically usable
  • No large overhead, integrated with legacy systems, etc…
thing Blink includes LedMsgs, TimerMsgs
{
  required port HW
  {
    sends ledToggle, timer_start
    receives timer_timeout
  }

  statechart BlinkImpl init Blinking
  {
    state Blinking
    {
      on entry HW!timer_start(1000)
      transition -> Blinking
      event HW?timer_timeout
      action HW!led_toggle()
    }
  }
}
link example and instance groups

```
configuration BlinkArduino
{
  group led : LedArduino
  set led.io.digital_output.pin = DigitalPin:PIN_13
  // The timer
  instance timer : TimerArduino
  // The blink application
  instance app : Blink
  connector app.HW => led.led.Led
  connector app.HW => timer.timer
}
```
The ThingML tools

Based on Eclipse / EMF Metamodel
Textual Syntax with EMFText
- For good usability and productivity
- To keep the development cost of the editor(s) reasonable

Graphical exports (graphML, graphviz, …)
Static well formedness and type checker
Equivalent compilers for a set of platforms
- C/C++ for different microcontrollers, linux, embedded linux
- Java for computers, smartphones, …
- Javascript (NodeJS)
- Maybe others if needed

Generators for communication channels
Easy to distribute ThingML IDE
```plaintext
thing Blink includes LedMsgs, TimerMsgs

required port HW
{
    sends led_toggle, timer_timeouts
    receives timer_timeouts
}

statechart BlinkImpl initial
{
    state Blinking
    on entry HW!timer_timeouts
    transition -> Blink
    event HW!timer_timeouts
    action HW!led_toggle
}

configuration Blink
{
    group led : Led
    set led.io.digital_out
    // The timer
    instance timer : SoftTimer
    // the blink application
}

Done uploading.
```

Binary sketch size: 3490 bytes (of a 250048 byte maximum)
Heterogeneous execution platforms

- how to handle deployment?
- how to handle reconfiguration?
- how to verify system level properties?

Effective support for runtime models

- limited resources
- time constraints

Distributed model dissemination
Models@runtime for distributed and heterogeneous systems
Kevoree in nutshell

- Lightweight M@R framework for building
- Distributed and Heterogeneous adaptive systems
- M@R Runtimes for
- Extensible Virtual System Infrastructure
- Resilient Software infrastructure based on diversity
Kevoree in nutshell
Kevoree Abstraction model for architectural diversity

Model@Runtime eases the management of distributed systems

Homogeneous adaptation for heterogeneous CPS devices

Large scale adaptation dissemination (Gossip, Paxos, etc.)

Ready for cloud and elasticity algorithms development

Open source

Kevoree in nutshell

WHAT IS KEVOREE?
Kevoree project aims at enabling distributed reconﬁgurable software development. Build around a component model, Kevoree leverages model@runtime approach to offer tools to build, adapt and synchronize distributed systems.

WHY KEVOREE?
Extensible, this project already offers runtime for JVM, Android, Arduino but also for virtualization management such as VirtualBox. In short, Kevoree helping you to develop your adaptable software from cloud stack to embedded devices!

💦 MAKE IT EASY
Kevoree offers an abstraction model that makes easier the manipulation of concepts such as Nodes, Components, Synchronization groups and Communication channels in distributed software systems.

💧 DISTRIBUTED MODELING
Modeling explicits the nodes and the topology of distributed systems. As a system representation, Kevoree allows the model itself to be distributed across the nodes to share the cluster organization.

นอกจากспеш
Build around asynchronous model, Kevoree allows design of asynchronous communication, asynchronous synchronization.

📢 OPEN SOURCE
Licensed under LGPL and maintained by the community on GitHub.

✍ CONTINUOUSLY DESIGN
Because long living system must evolve while serving, our approach takes care about HotSwap functionality.

☁️ BUILD YOUR OWN CLOUD
Particular software always need particular optimization. Embedding your own adaptation layer in Kevoree cluster allows you to build your own cloud.

NEWS

**Kevoree 3.0.5 out !**
Two news, ﬁrst of all the ﬁrst version of Kevoree 3.XX branch is out with version number 3.0.5 and respectively 3.0.2 for base library. In addition we will be present in China @Middleware conference to present it. See you at Beijing !

**KMF 1.7.0**
KMF 1.7.0 is now released ! With many new features for aspects oriented develempent and better scalability for JS platform. Two new samples are coming to explain how to build an ecore web editor with KMF... stay tuned... Duke :-(

DECEMBER 7, 2013

OCTOBER 13, 2013
Current Works
Automatic synthesis of software diversity

- sosies are different programs that still belong to the correctness envelop
Automatic diversity synthesis

Automatic synthesis of software diversity

- sosies are different programs that still belong to the correctness envelop
Software intensive systems

Continuous design for heterogeneous, variable, adaptive within reasonable correctness boundaries.