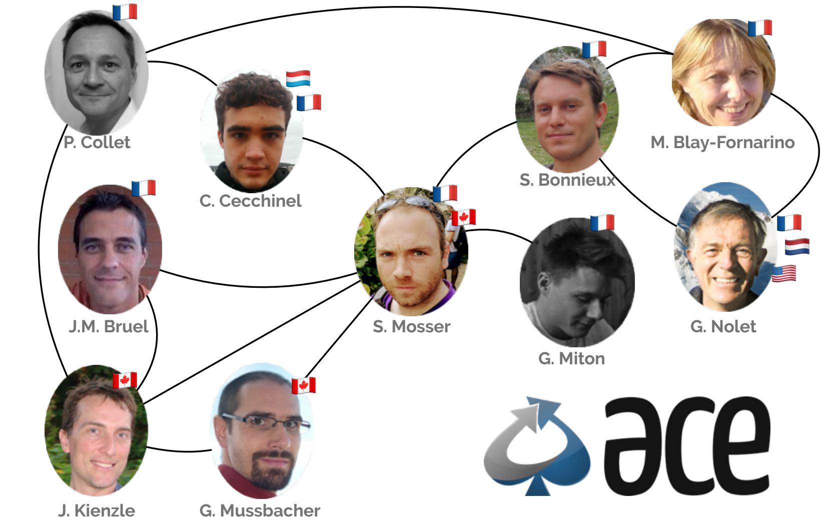
Software Composition
in a Cyber-Physical World

Sébastien Mosser
Ptidej seminar at Concordia University
13.12.2019

UQÀM | Département d'informatique

This is a team effort, started in 2014!



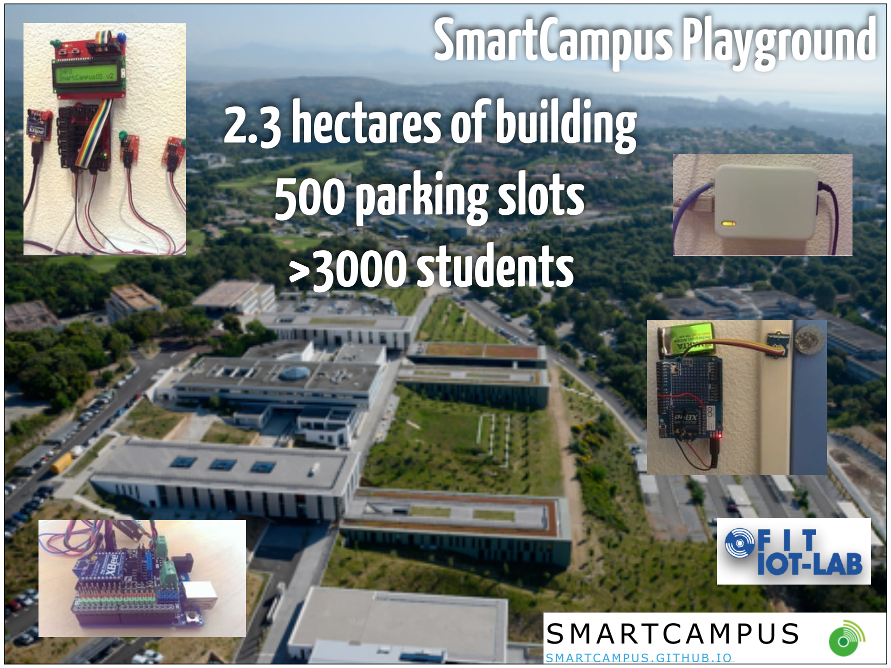
Research Challenge

How to tame the complexity of designing complex applications for Cyber-Physical Systems ?

Previous Work (Cyril Cecchinel's PhD)

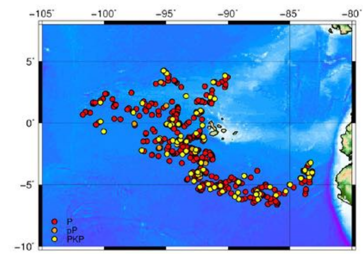
- SmartCampus, a sensor network for experiments *[SERVICES'14]*
- Shared Sensing Infrastructure *[ICSR'16]*
- Composable Workflows on top os sensor networks *[SAC'16]*
- Automated deployment *[APSEC'16]*
- DEPOSIT reference implementation *[PhD'17]*
- Machine learning for sensor data prediction *[FGS'19]*
  - *Industrial collaboration with DataThings*

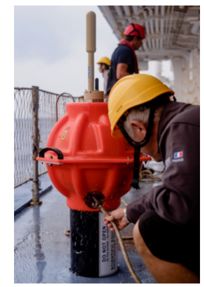
Cyril was a PhD student in the group (2014-2017). He is now lead research engineer at DataThing (Luxembourg), a spinoff of the SnT research center that develops the GreyCat database engine for large-scale sensor networks.

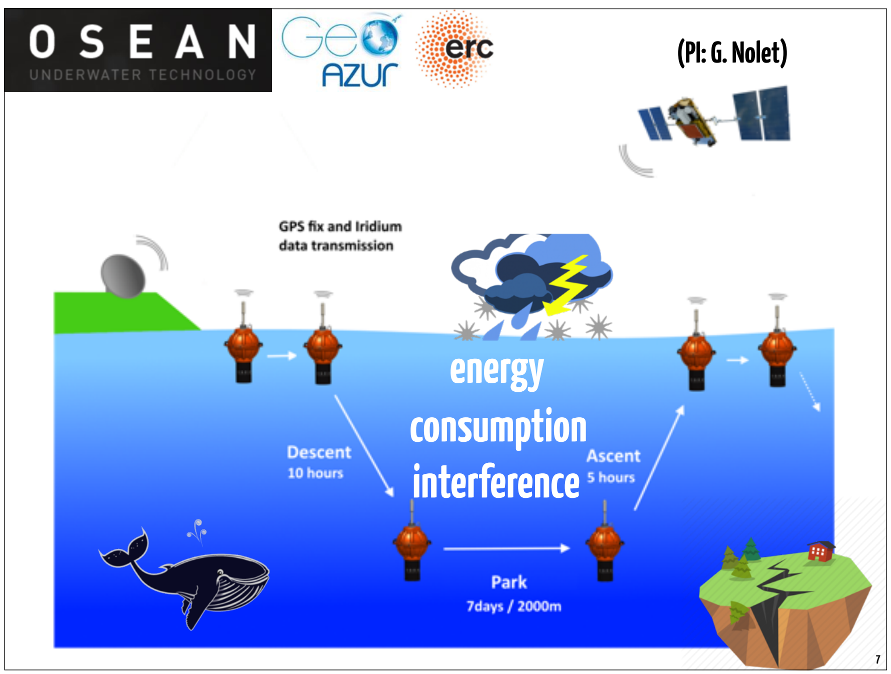


Ongoing work : the MERMAID project

- Sébastien Bonnieux's PhD Thesis (2017-…)
  - collaboration with the Geoscience institute in Nice (FR) *[OCEANS'19]*
  - Using oceans to monitor earthquakes, and other "stuff"
- Research challenge: allow the scientist to understand what is happening at the code level







Agenda

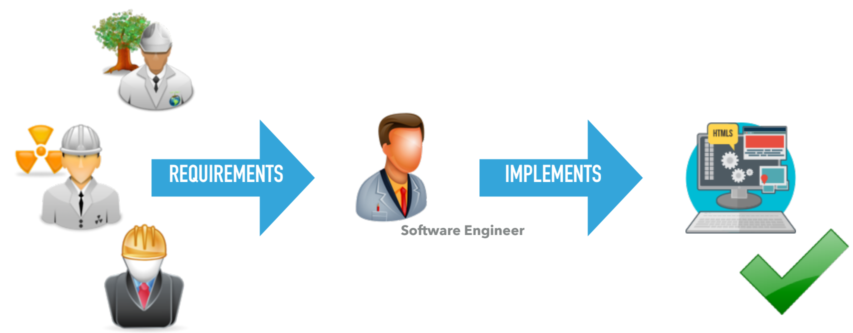
Automated deployment of data collection policies at large scale*

** I had to made choices … and this contributions covers several dimension of the work!*

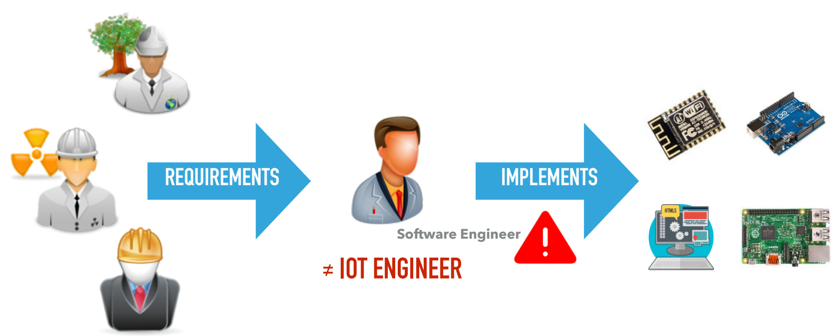
C. CECCHINEL, S. MOSSER. P. COLLET

AUTOMATED DEPLOYMENT OF DATA COLLECTION POLICIES OVER HETEROGENEOUS SHARED SENSING INFRASTRUCTURES

TRADITIONAL SOFTWARE DEVELOPMENT



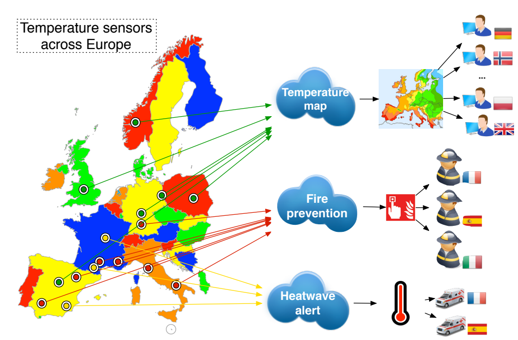
IOT DEVELOPMENT



SENSOR NETWORKS CONFIGURED AT THE HARDWARE LEVEL
LOW-LEVEL PROGRAMMING LANGUAGES

→ TEDIOUS AND ERROR-PRONE ACTIVITIES

AD-HOC NETWORKS



AD-HOC NETWORKS
NO SHARING
DATASET ISOLATED

→ DIFFICULT TO (RE-)USE

« The most obvious drawback of the **current WSNs** is that they are **domain-specific** and task-oriented, tailored for particular applications with **little or no possibility of reusing them** for newer applications »

« This strategy leads to **redundant deployments** when new applications are contemplated »

Khan, I., Belqasmi, F., Glitho, R., Crespi, N., Morrow, M., & Polakos, P. (2015). Wireless Sensor Network Virtualization: A Survey.

CONTRIBUTION

- A toolchain that supports:
  - **High-level** data collection **policies**
  - **Platforms** and **network representations**
  - **Composition** and **deployment** over heterogeneous sensing infrastructures

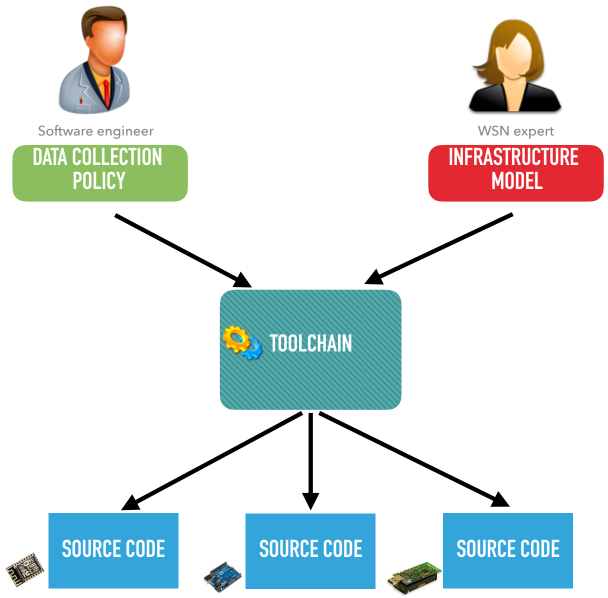
FULLY AUTOMATED APPROACH

REQUIREMENTS

- Separation of concerns
- Automatic tailoring of policies
- Automatic projection of policies
- Automatic sharing

REQUIREMENTS

- Separation of concerns
- Automatic tailoring of policies
- Automatic projection of policies
- Automatic sharing



DATA COLLECTION POLICY


Software engineer

« a set of operations performed on data to convert them into knowledge » [1]

Must be **abstracted** enough to let the **software engineer** focused on her business activities

workflow

/ˈwərkˌflou/

« a **sequence** of **activities** performed in a business that produces a result of **observable** value to an individual actor of the **business** » [2]

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 2013.

[2] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature- Oriented Domain Analysis (FODA). Technical Report CMU/SEI-90-TR-21, 1990.

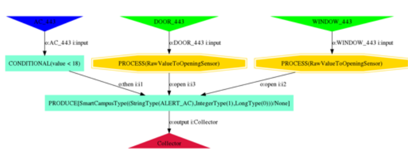
DATA COLLECTION POLICY


Software engineer

- A domain-specific language to define Data Collection Policies

```
flows {
  ac_443() -> temp_filter("input")
  door_443() -> door_converter("input")
  window_443() -> window_converter("input")
  temp_filter("then") -> produce("i1")
  window_converter("open") -> produce("i2")
  door_converter("open") -> produce("i3")
  produce("output") -> collector()
}
```

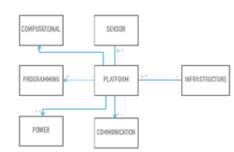
- Sensor/Collectors declaration
- Activity definition
- Data-flows definition

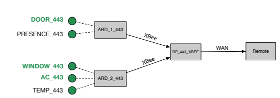


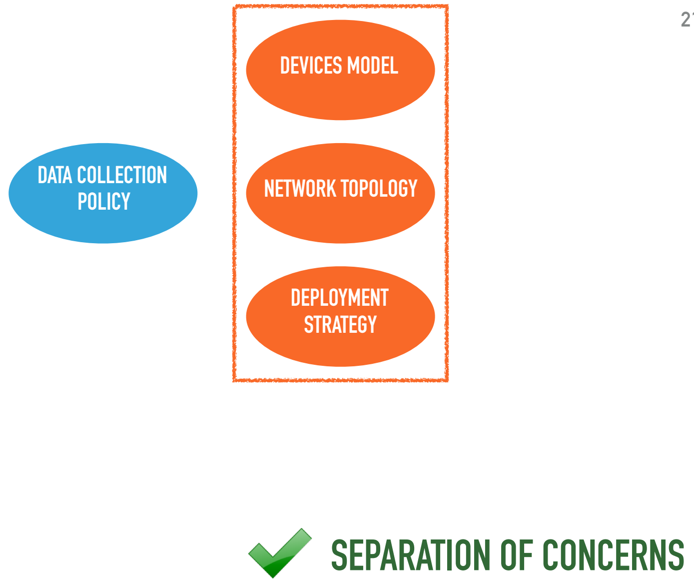
INFRASTRUCTURE MODEL

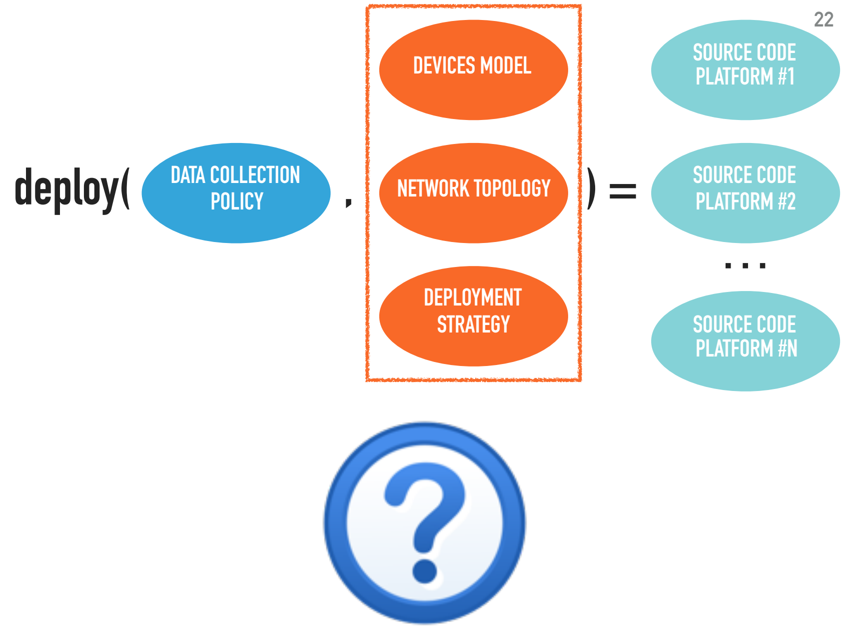
Three models to describe the sensing infrastructure

- **Platform variability** model
- **Network topology** model
- **Deployment strategy** model





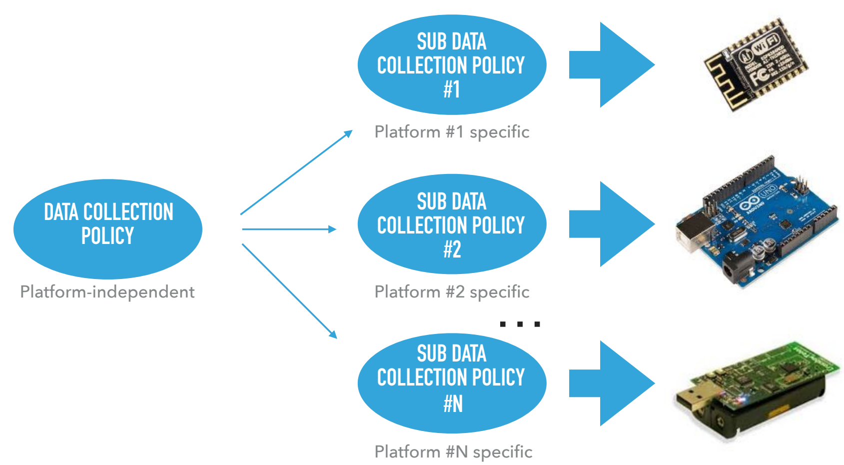




REQUIREMENTS

- Separation of concerns
- Automatic tailoring of policies
- Automatic projection of policies
- Automatic sharing

Build **platform specific** data collection policies from a **platform independent** policy



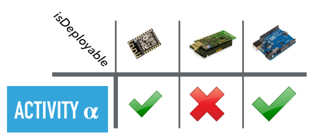
Decomposition

Two operators defined at the **data collection policy layer**:

`req` returns the subset of sensors needed for the realization of the activity

req(ACTIVITY α) = { S2 ; S3 }

`isDeployable` check if an activity is deployable on a given platform



Decomposition

An operator defined at the **network topology layer**:

`reach` returns the subset of sensors reachable from a given platform

reach() = {S1 ; S2 ; S3 }

platform #1

Decomposition

An operator defined at the **deployment strategy layer**:

`place` For a set of platforms, return the platform that maximize the strategy's objectives

place(ACTIVITY α , {P1;P2;P3}) = P1

Decomposition

For each activity a, find platforms p satisfying the property:

`req(a) ⊂ reach(p) ∧ isDeployable(a, p)`

If no platform satisfies the property, an error is returned to the software engineer

ACTIVITY α ··········> `{platform #1, platform #3, platform #4}`

ACTIVITY β ··········> `{platform #2, platform #4}`

...

ACTIVITY ω ··········> `{platform #3}`

Decomposition

Use the place operator to select the appropriate target platform among the available candidates

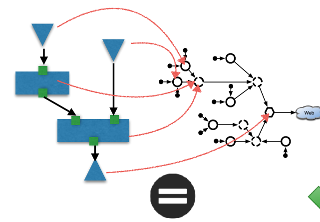
place(ACTIVITY α , `{platform #1, platform #3, platform #4}`) = platform #1

place(ACTIVITY β , `{platform #2, platform #4}`) = platform #4

. . .

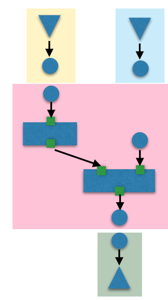
place(ACTIVITY ω , `{platform #3}`) = platform #3

Decomposition



AUTOMATIC TAILORING OF POLICIES

AUTOMATIC PROJECTION OF POLICIES



Routing

on **each** platform **not involved** in the process

WHAT IF A POLICY HAS ALREADY BEEN DEPLOYED ON THE TARGETED PLATFORM ?

REQUIREMENTS

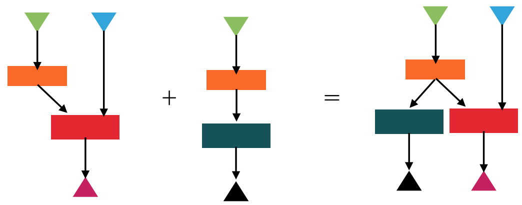
- Separation of concerns
- Automatic tailoring of policies
- Automatic projection of policies
- Automatic sharing

Composition

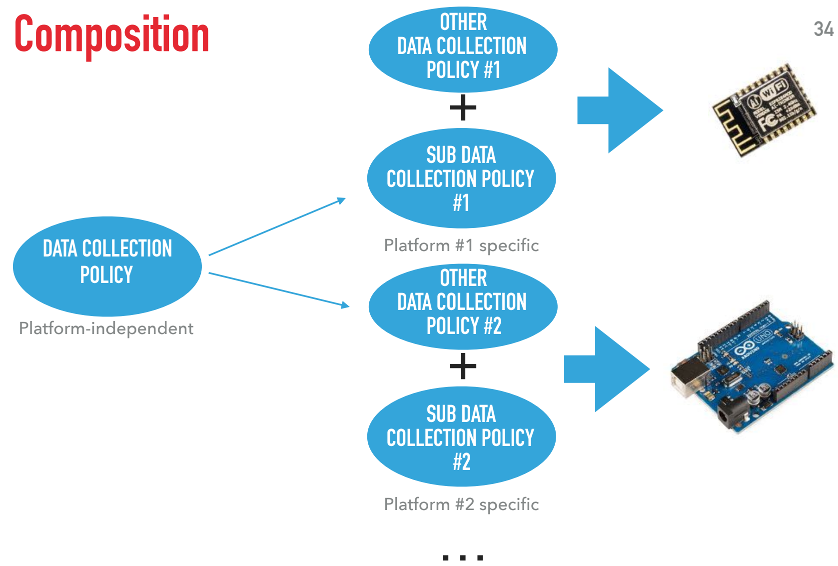
An operator defined at the **platform-specific data collection policy layer**:

\+ compose two policies together

Extension of the graph series composition

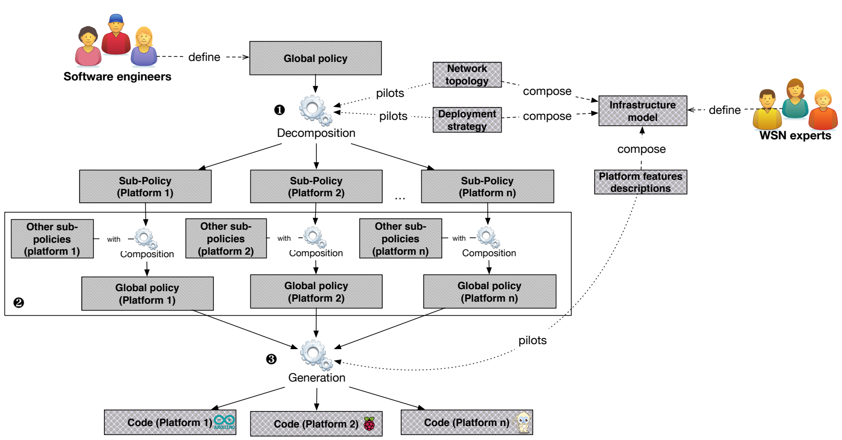


Composition



AUTOMATIC COMPOSITION

TOOLCHAIN IN ONE SLIDE



[ASSESSMENT]

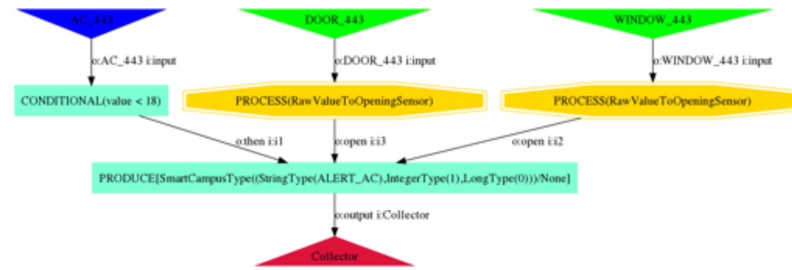
DEPOSIT

Data collEction POlicies for Sensing InfrasTructures

Open-source toolchain available on Github
https://github.com/ace-design/DEPOSIT

DATA COLLECTION POLICY

- As a software engineer, I would like to receive AC data if the door and the window are opened for office 443 to monitor the energy loss



VALIDATION CRITERIA

- Separation of concerns:
  - Design using only **activities**
  - Deployment **without** a-priori knowledge
- Automatic tailoring of policies
  - Generated code should call the **right** libraries
- Automatic projection of policies
  - Activities are projected on the **appropriate** platform
  - **Ready-to-flash** code

USING THE TOOLCHAIN

DEPLOYMENT OF THE RUNNING EXAMPLE (COMPREHENSIVE POLICY: 50 OFFICES)

	DEPOSIT source	# Generated files	# Generated LoC	# Concepts (before expansion)	# Concepts (after expansion)	Deployment time (in s)
Template	19	N/A	N/A	N/A	N/A	N/A
Single office	19	3	267	5	8	2.5
Comprehensive policy (without composition)	455	105	11685	250	400	50

ALERT_AC2_ARD_2_443_1467106993529.ino
ALERT_AC2_ARD_2_444_1467106075134.ino
ALERT_AC2_ARD_2_445_1467106061773.ino
ALERT_AC2_ARD_2_446_1467101805650.ino
ALERT_AC2_ARD_2_447_1467101803776.ino

```
#include <grovetemperature.h>
#include <raw.h>

#define BOARD_ID "ARD_2_443"
```

We consider **15 minutes** as the required time for a network expert to write and enact the code for a given office without using any aspect of the toolchain

USING THE TOOLCHAIN

- Automatic sharing
  - ☆ Successful deployment of multiple applications

App #1: Air conditioning warning

App #2: Fahrenheit converter

App #3: Parking space occupancy

100 OFFICES

250 PARKING SPACES

BLANK INFRASTRUCTURE

ONE BORDER-ROUTER

USING THE TOOLCHAIN

Deployment of App #1 - no composition triggered

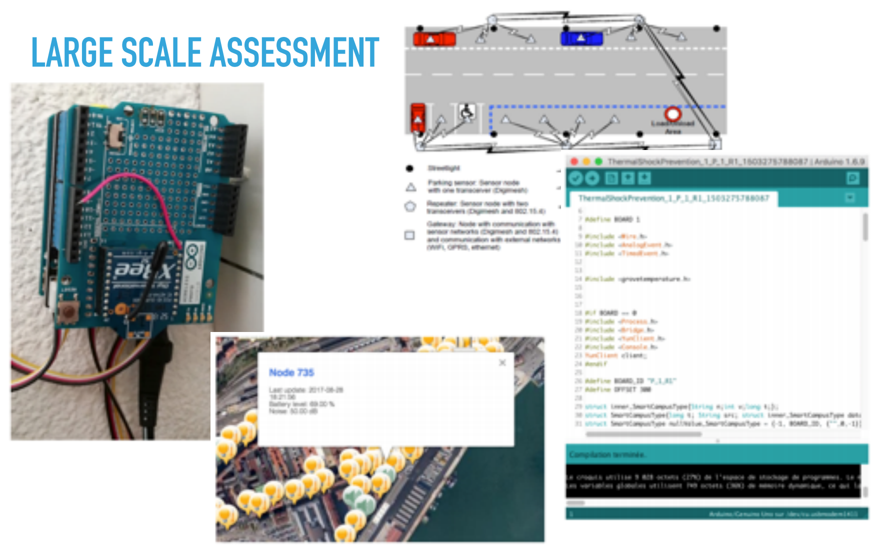
Deployment of App #2 - 101 compositions triggered

Deployment of App #3 - 1 composition triggered

[LARGE-SCALE ASSESSMENT]

HERE "LARGE" MEANS "REALISTIC"

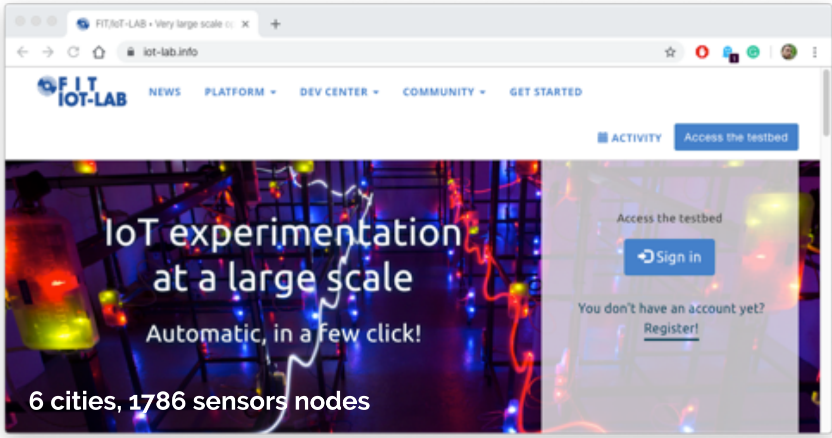
LARGE SCALE ASSESSMENT



Topology to simulate SmartSantander deployment

FIT IOT LAB

French national platform for IoT experiments



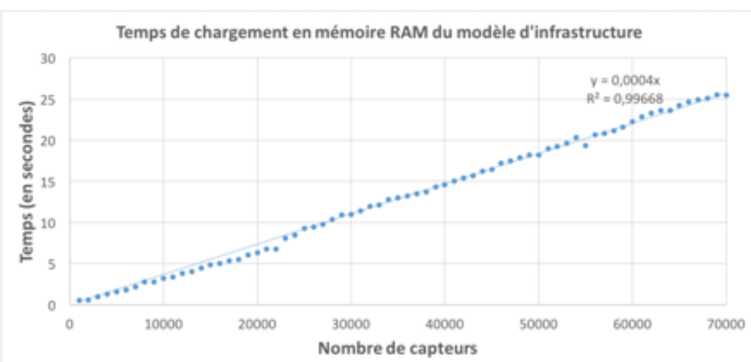
Think "Compute Canada", but for the IoT.

METHODOLOGY

1. Select a use-case from an existing SmartCity / Building
2. Deploy a prototype on SmartCampus (up to 50 sensors)
3. Scale-up with FIT IoT Lab (up to 500 sensors)
4. Use simulation to scale-up to 60,000+ sensors

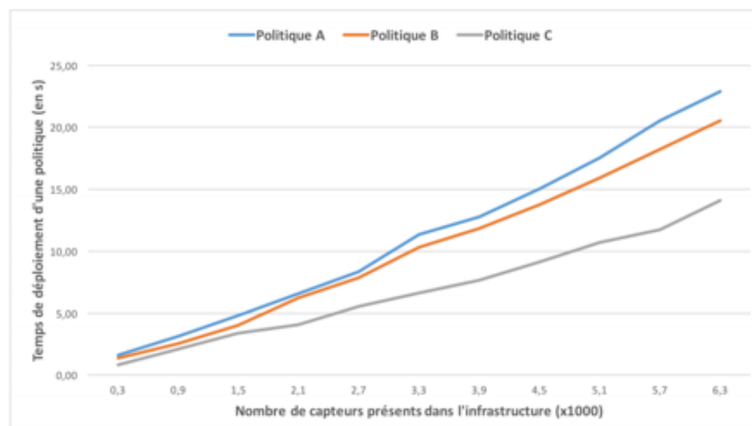
Step 1 based on existing literature, platforms & documentation
Steps 2 & 3 deployed on real hardware
Step 4 extrapolate from the previous results

LINEAR ANALYSIS OF THE TOPOLOGY



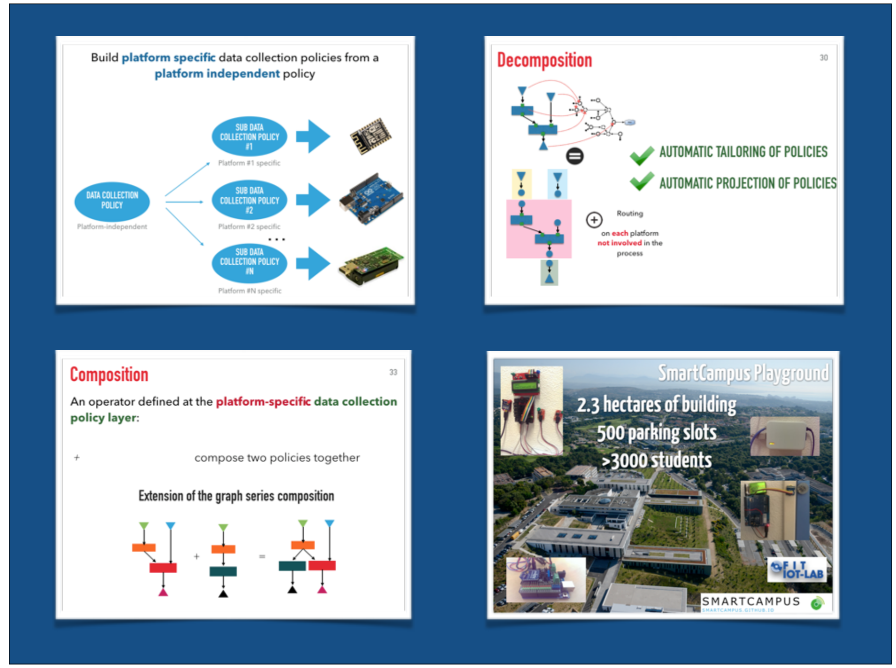
"In practice, it works"

LINEAR DEPLOYMENT TIME OF A POLICY



"In practice, it works"

[CONCLUSIONS]



PERSPECTIVES (FRQNT TEAM PROJECT, UNDER REVIEW)

Software Modelling for Constrained Environments with Domain Experts in the loop

Explainable composition & human-in-the-loop optimization