Confirm
Smart Manufacturing

Confirm
Smart Manufacturing

Accelerating Application Development in the Internet of Things using Model-driven Development

Pankesh Patel

CONFIRM is the Science Foundation Ireland Research Centre for Smart Manufacturing
Transforming Irish industry to become **world-leaders in smart manufacturing**

A World Leading SFI Research Centre

Science Foundation Ireland SFI For what's next

University of Limerick Ollscoil Luimnigh · Tyndall National Institute · UCC University College Cork, Ireland Coláiste na hOllscoile Corcaigh · NUI Maynooth Ollscoil na hÉireann Má Nuad · OÉ Gaillimh NUI Galway · CIT Cork Institute of Technology Institiúid Teicneolaíochta Chorcaí · AIT · LIT

Outline…

- Research projects
  - IoTSuite – A Toolkit for Prototyping Internet of Things Applications
    - To enable IoT application development with minimal development effort
    - Funded by: INRIA Paris – Rocquencourt, France (2010 – 2014)
  - SMEWB – Subject Matter Expert Workbench
    - To create, reuse and deploy analytic algorithms in ABB products and solutions with little or no additional coding
    - Subject Matter Experts (SMEs) – domain expertise but very little programming experience
    - Funded by: ABB Corporate Research (2014 – 2017)

Basic building blocks of "Internet of Things"

Things have sensors attached,
Things have actuators attached
Things have storage attached,
Things may have computational capabilities

1

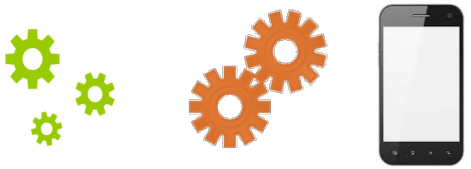
Physical & virtual devices (or "things")

The produced information is communicated over network using various protocols

2

Communication

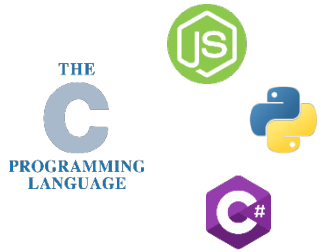
People & Process



3

These networked inputs can then be combined into a system that integrate data, people (i.e., user interactions), processes to achieve a specific objectives.

Programming in IoT



Programming Languages

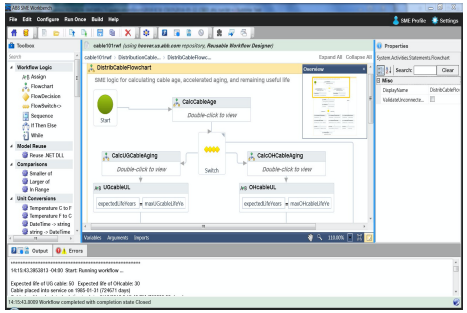
- Full control on AL (app logic)
- More development effort

Rapid prototyping tools

- Reduce development effort
- Platform-specific design (Language, Runtime)

Cloud

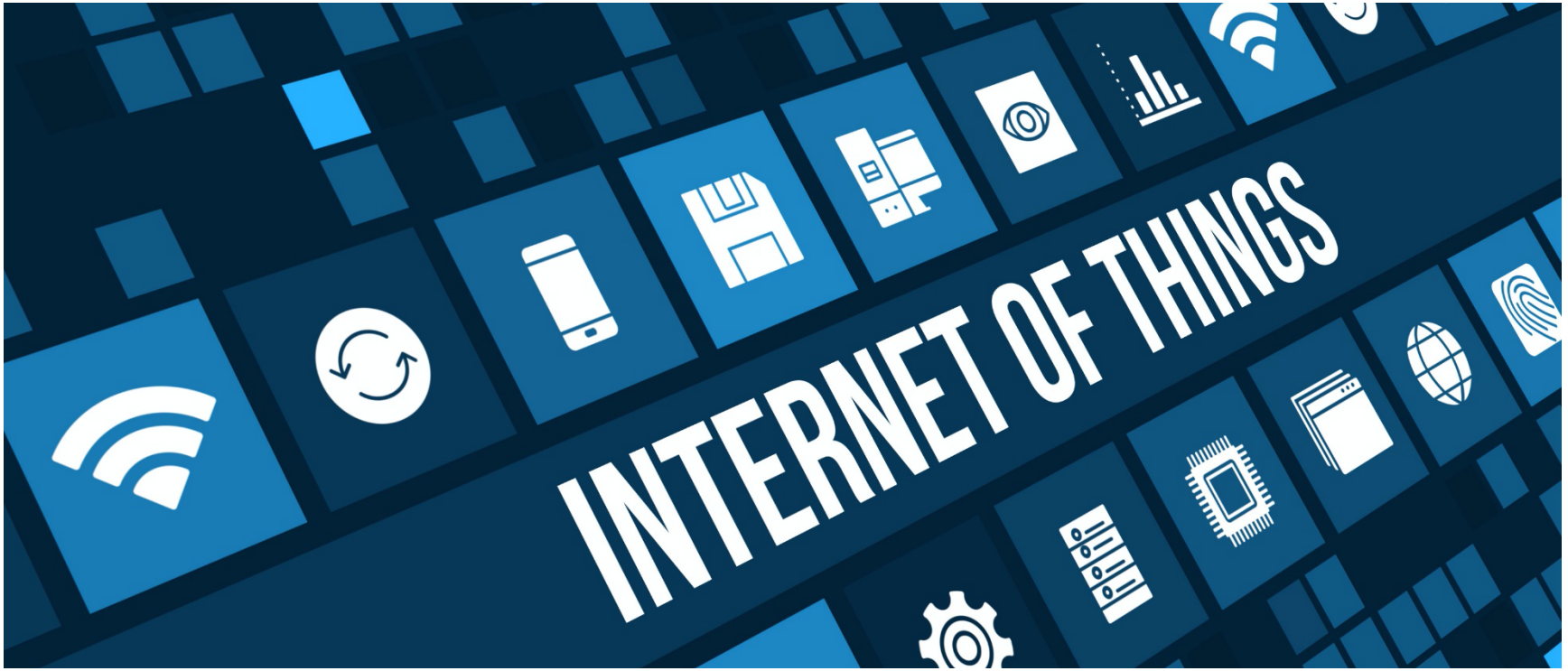
- Reduce development effort, ease of deployment & evolution (due to centralized system)
- Cloud-dependent design



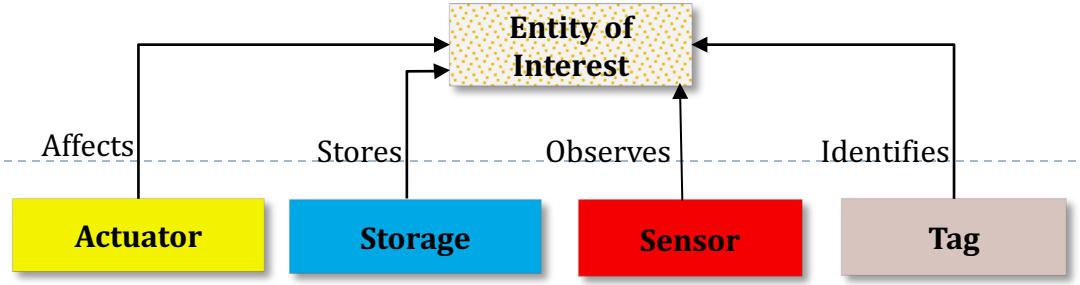
MDD

- PIM, PSM

IoTSuite: A Toolkit for Prototyping IoT Applications



Model



Things (or *Entity of Interest*) is an object, including attributes that describe it, and its state that is relevant from a user or an application perspective.

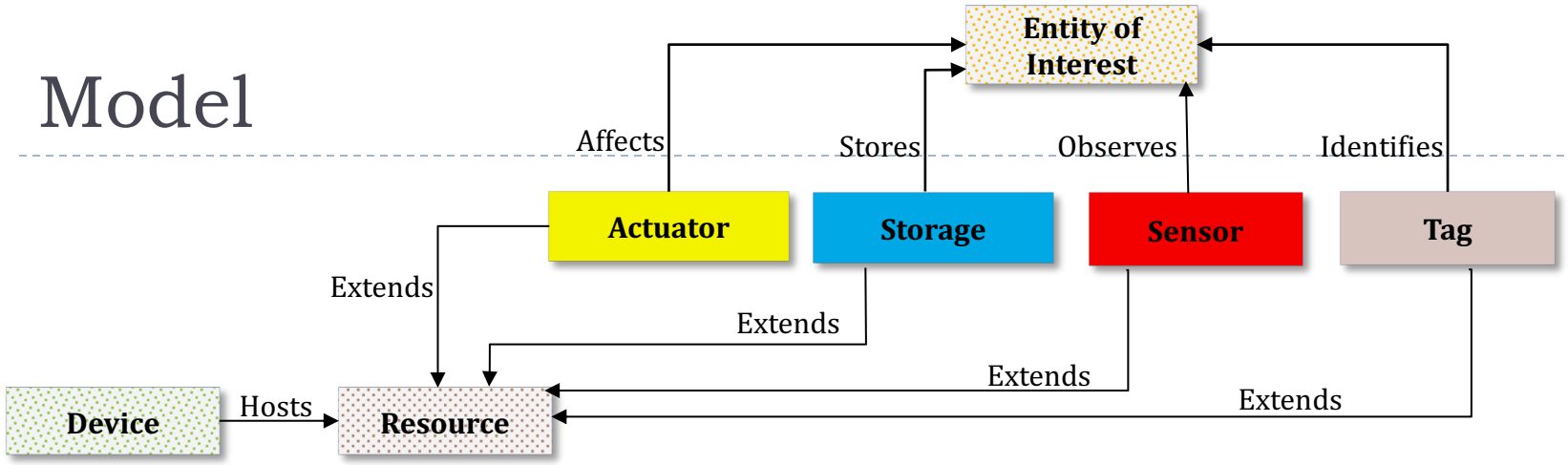
Heart rate

Temperature value of a **Room**

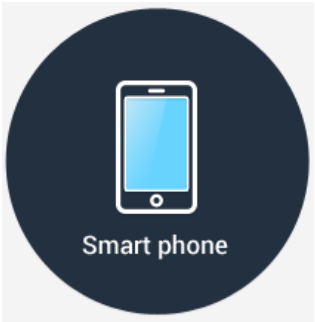
Battery level of a **Smartphone**

- Temperature sensor observes the temperature value of a room
- Heater controls the temperature of a room
- Profile Database stores user's various preferences
- Badge identifies a person.

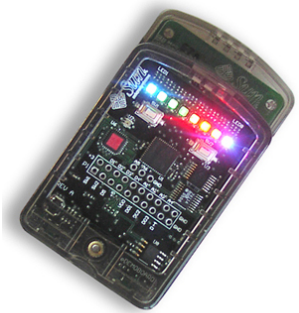
Model



- A resource is an abstract representation of a sensor, an actuator, a storage
- A device is an entity that provides resources the ability of interacting with other devices.



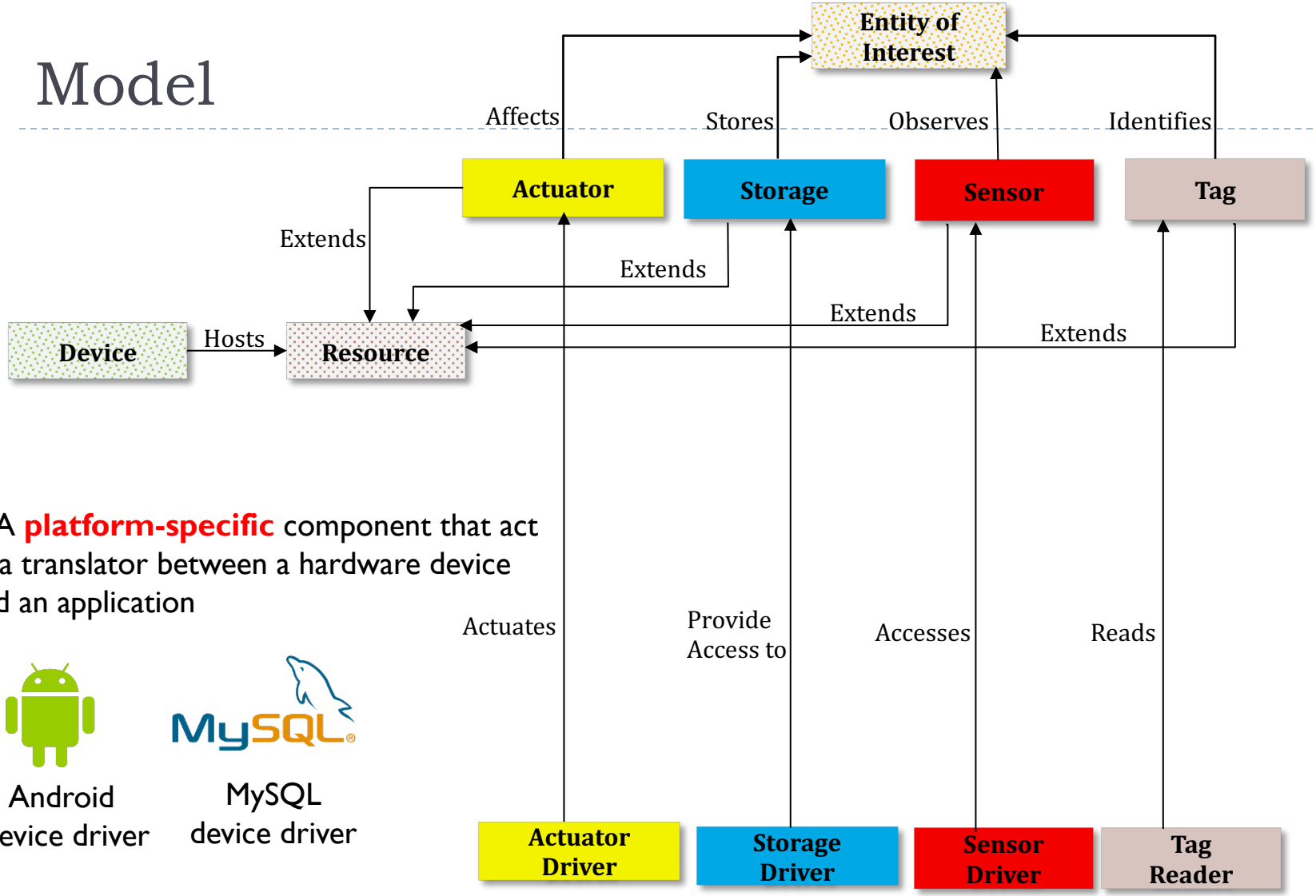
Smart Phones



SunSPOT

Cloud server

Model

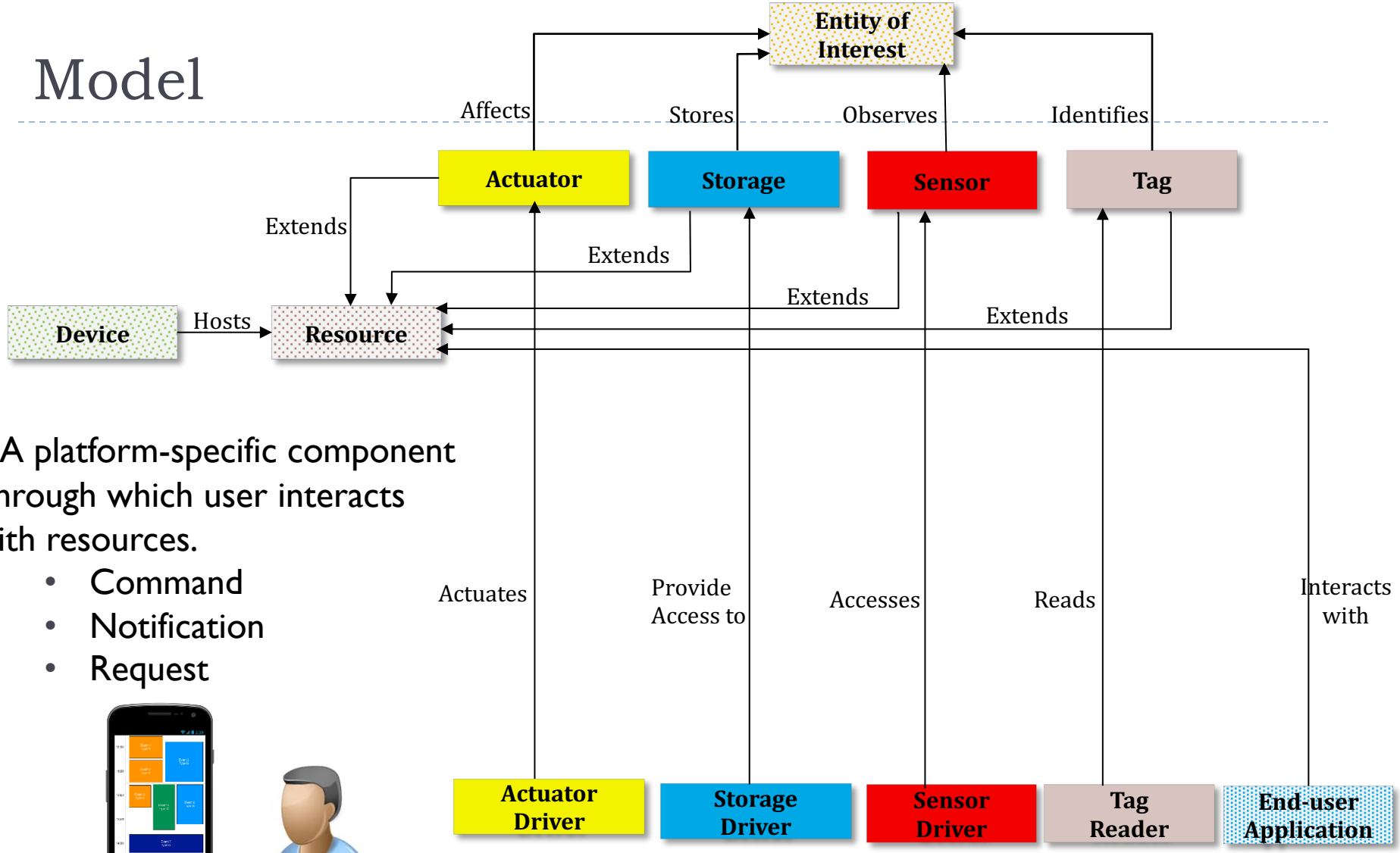


- A **platform-specific** component that act as a translator between a hardware device and an application

Android device driver

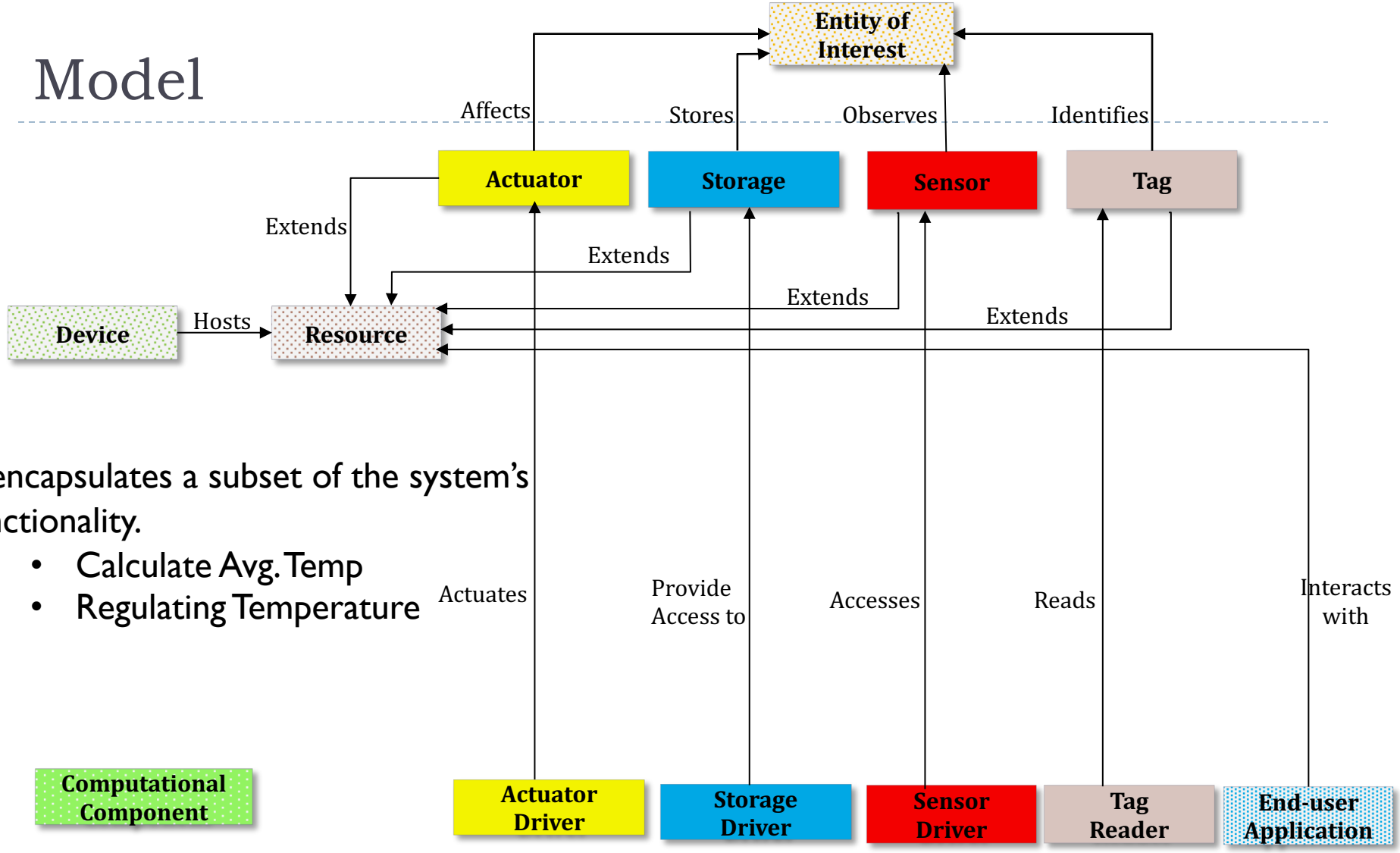
MySQL device driver

Model



- A platform-specific component Through which user interacts with resources.
  - Command
  - Notification
  - Request

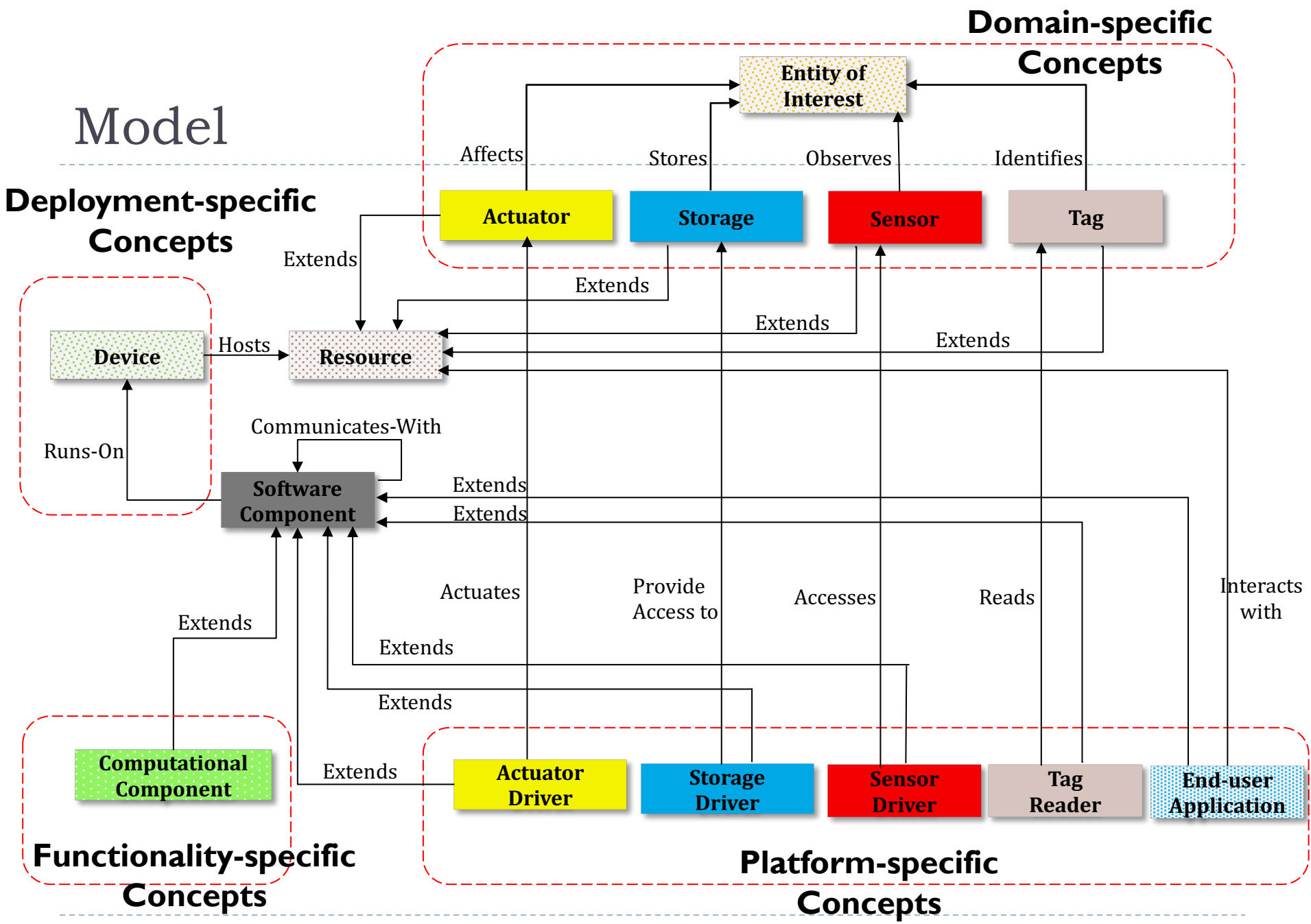
Model



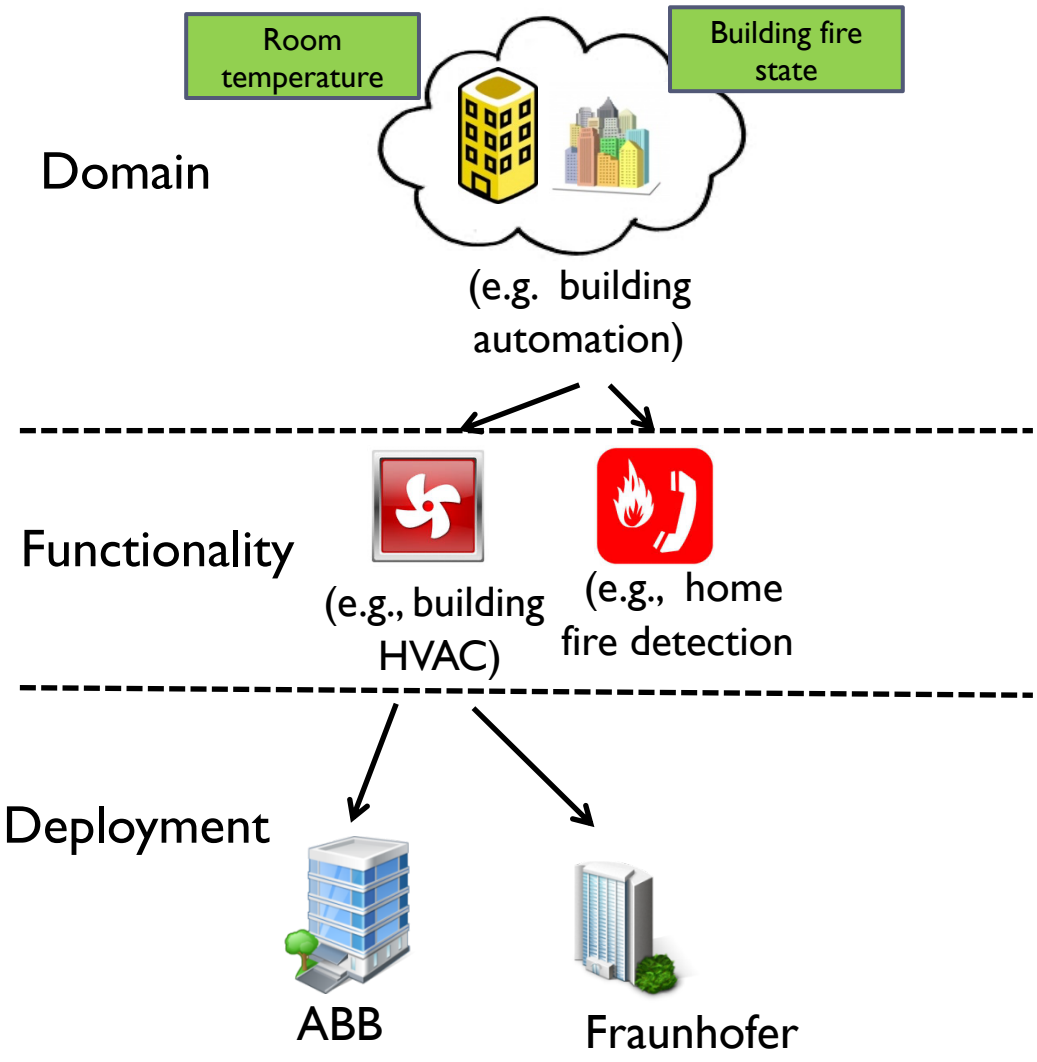
It encapsulates a subset of the system's functionality.

- Calculate Avg. Temp
- Regulating Temperature

Model

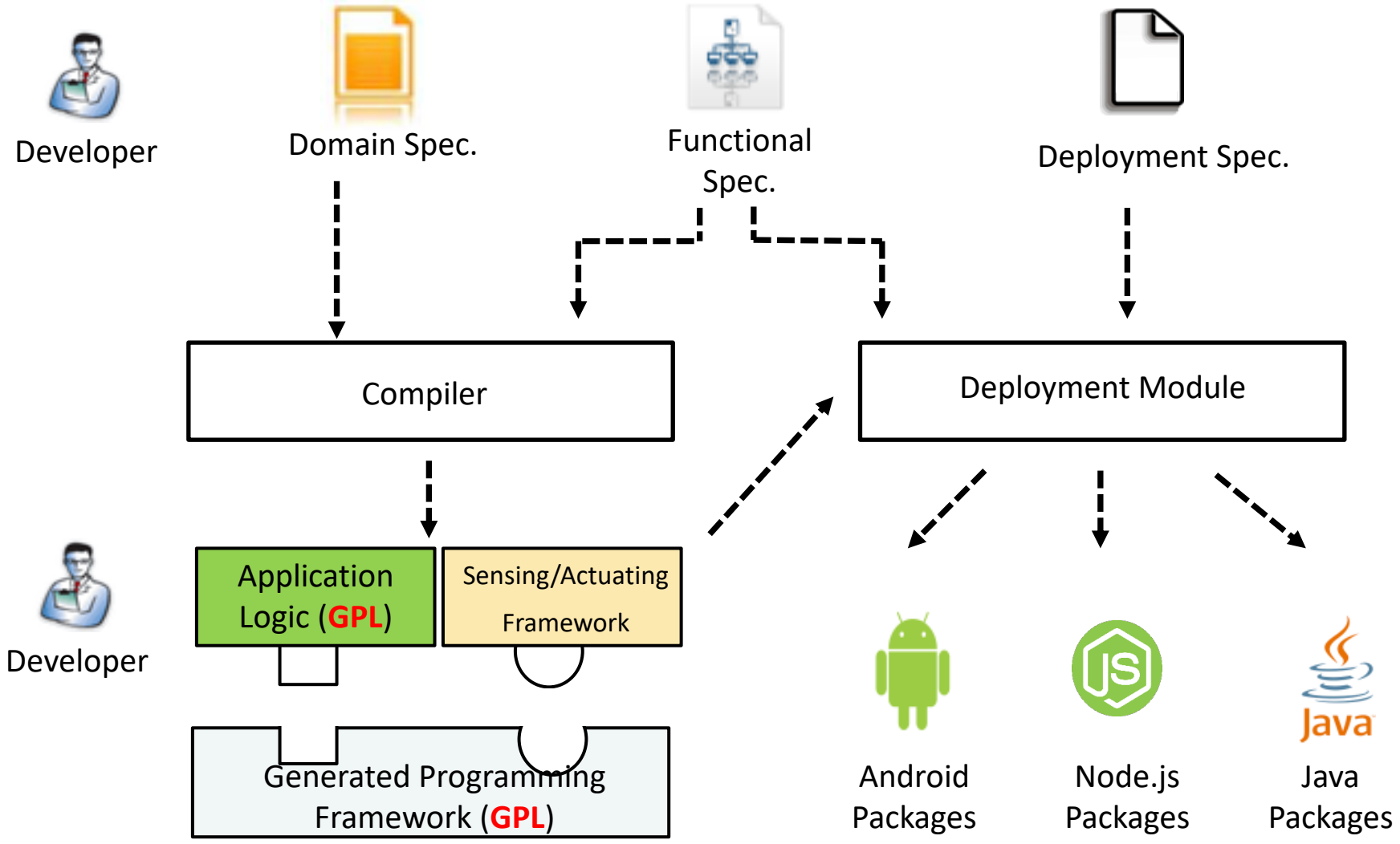

Domain-specific Concepts
Entity of Interest
Affects
Stores
Observes
Identifies
Actuator
Storage
Sensor
Tag
Deployment-specific Concepts
Extends
Extends
Extends
Extends
Device
Hosts
Resource
Communicates-With
Runs-On
Software Component
Extends
Extends
Actuates
Provide Access to
Accesses
Reads
Interacts with
Extends
Extends
Extends
Computational Component
Extends
Actuator Driver
Storage Driver
Sensor Driver
Tag Reader
End-user Application
Functionality-specific Concepts
Platform-specific Concepts

Separation of Concerns

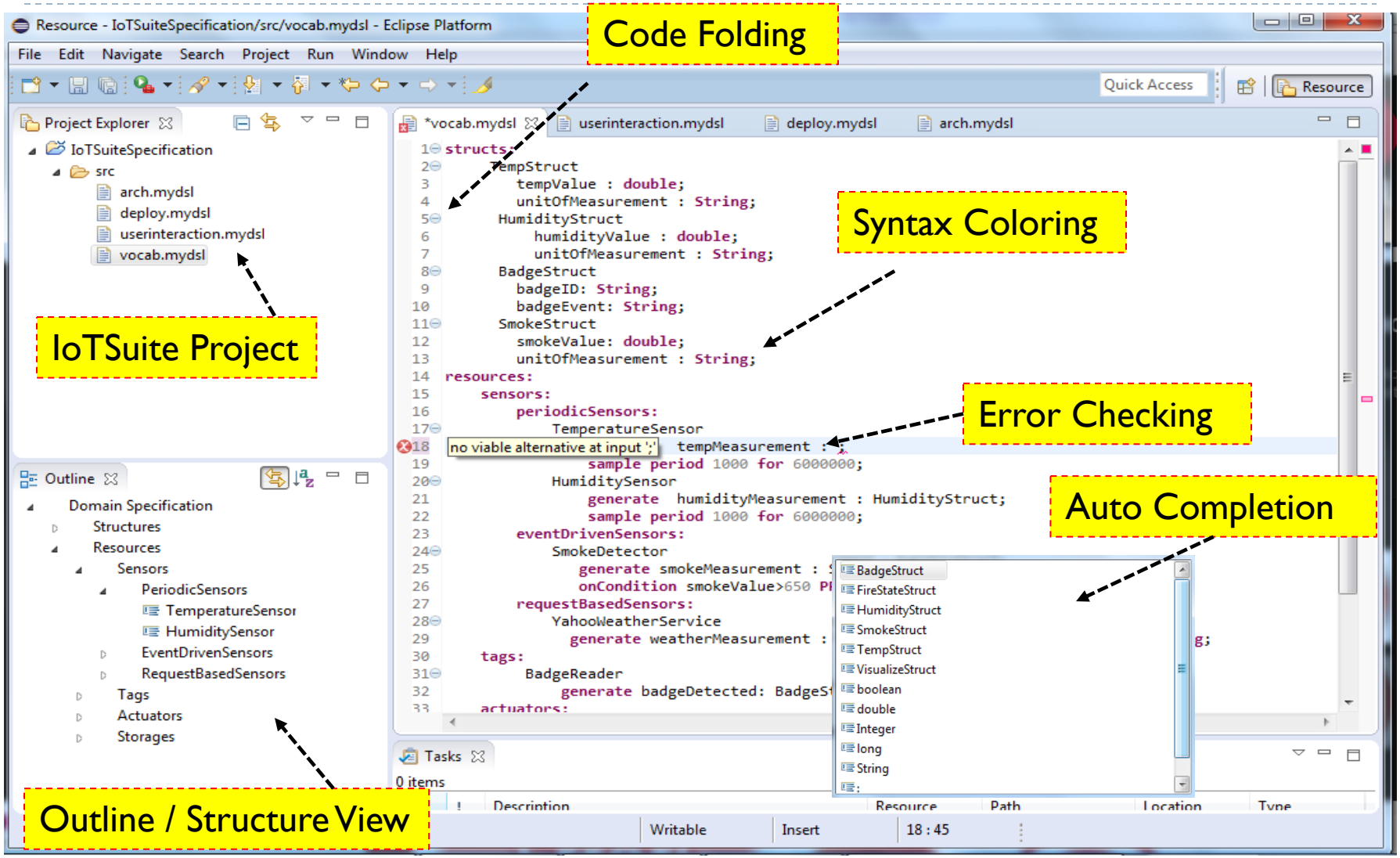


"Reusability across concerns"

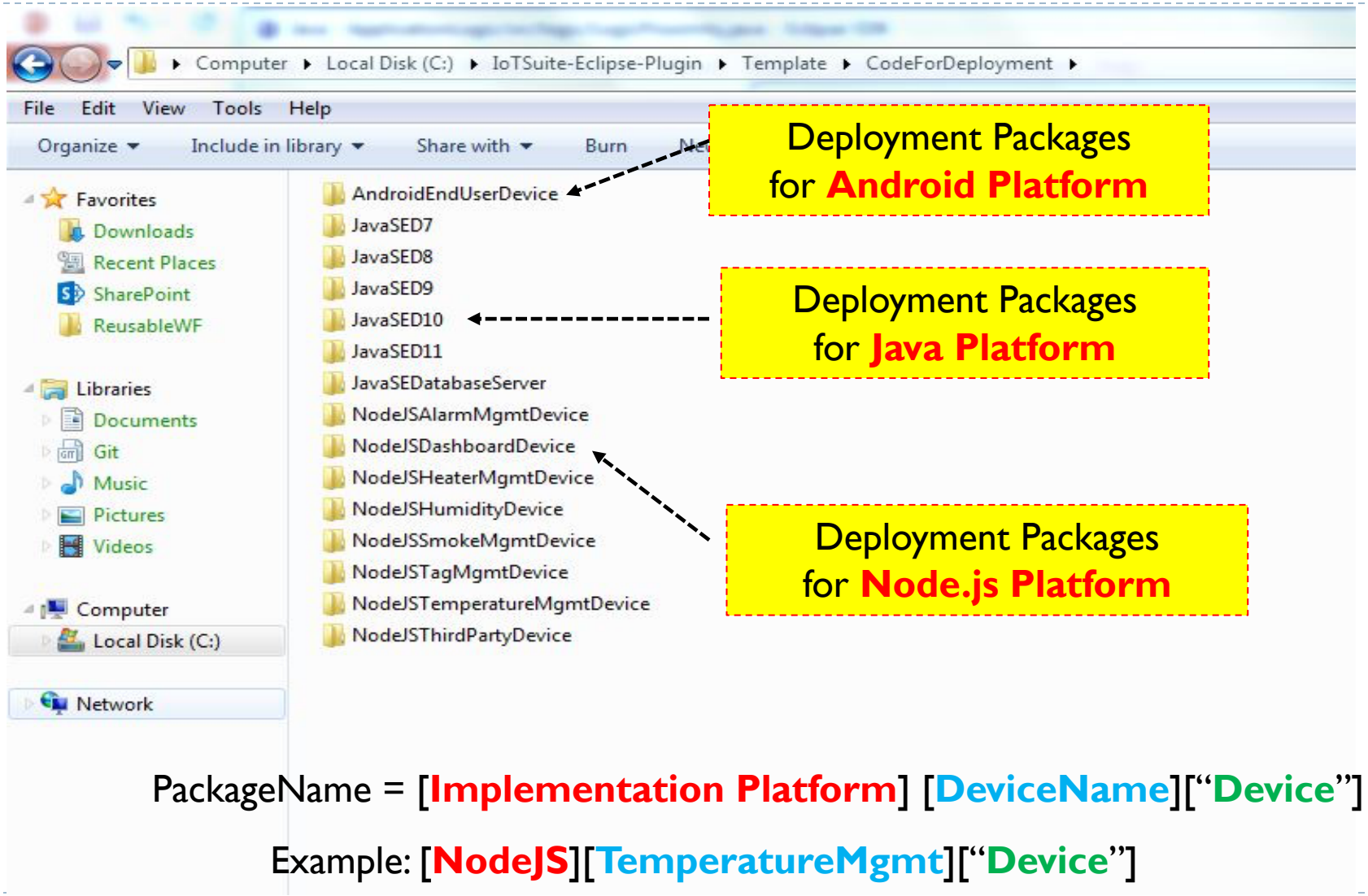
IoTSuite: Overview



Code editor


Code Folding
Syntax Coloring
IoTSuite Project
Error Checking
Auto Completion
Outline / Structure View
Resource - IoTSuiteSpecification/src/vocab.mydsl - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Quick Access
Resource
Project Explorer
IoTSuiteSpecification
src
arch.mydsl
deploy.mydsl
userinteraction.mydsl
vocab.mydsl
*vocab.mydsl
userinteraction.mydsl
deploy.mydsl
arch.mydsl
structs:
TempStruct
tempValue : double;
unitOfMeasurement : String;
HumidityStruct
humidityValue : double;
unitOfMeasurement : String;
BadgeStruct
badgeID: String;
badgeEvent: String;
SmokeStruct
smokeValue: double;
unitOfMeasurement : String;
resources:
sensors:
periodicSensors:
TemperatureSensor
no viable alternative at input ';'
tempMeasurement :
sample period 1000 for 6000000;
HumiditySensor
generate humidityMeasurement : HumidityStruct;
sample period 1000 for 6000000;
eventDrivenSensors:
SmokeDetector
generate smokeMeasurement :
onCondition smokeValue>650 PI
requestBasedSensors:
YahooWeatherService
generate weatherMeasurement :
tags:
BadgeReader
generate badgeDetected: BadgeSt
actuators:
Outline
Domain Specification
Structures
Resources
Sensors
PeriodicSensors
TemperatureSensor
HumiditySensor
EventDrivenSensors
RequestBasedSensors
Tags
Actuators
Storages
BadgeStruct
FireStateStruct
HumidityStruct
SmokeStruct
TempStruct
VisualizeStruct
boolean
double
Integer
long
String
Tasks
0 items
Description
Resource
Path
Location
Type
Writable
Insert
18 : 45

IoTSuite deployment packages

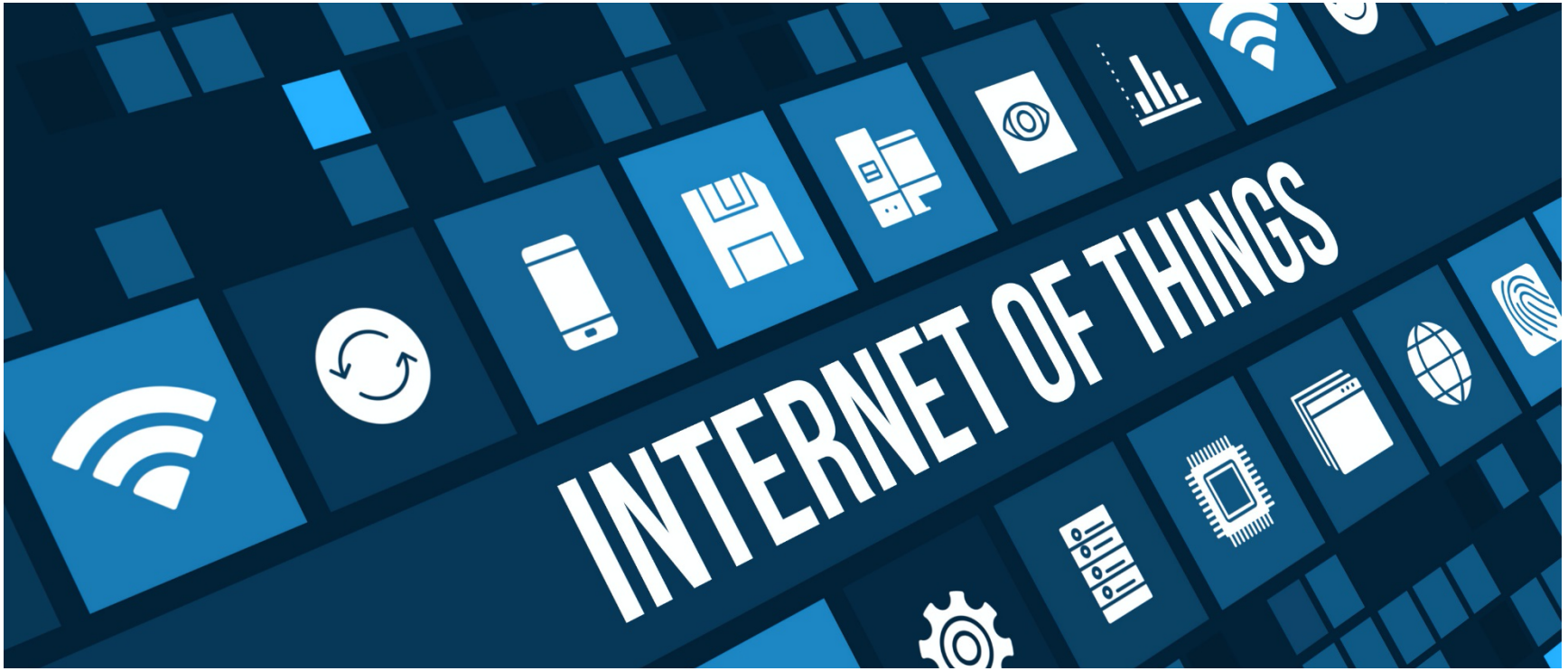


PackageName = [**Implementation Platform**] [**DeviceName**][“**Device**”]

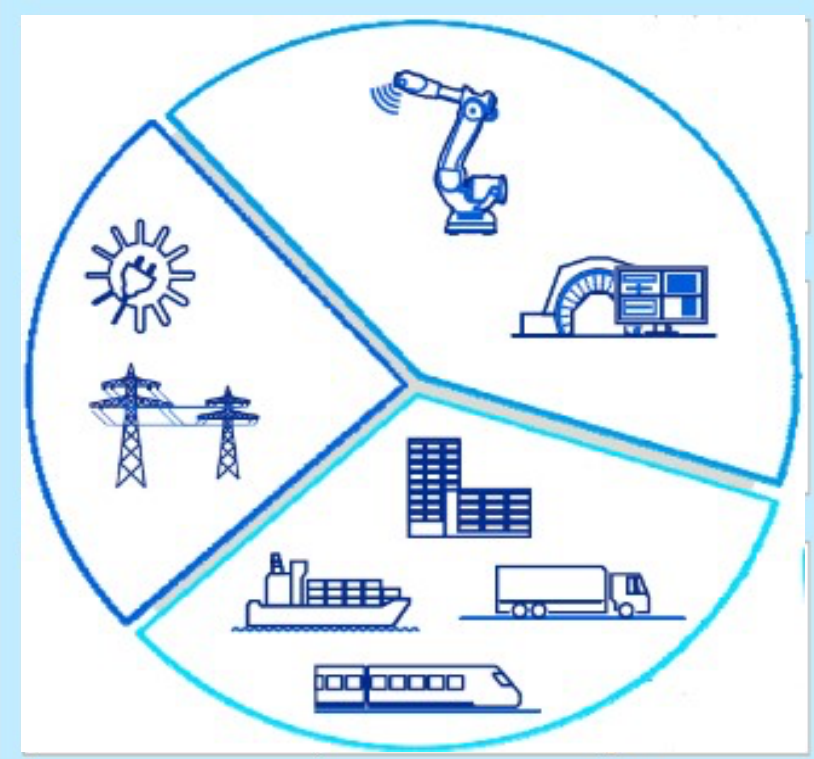
Example: [**NodeJS**][**TemperatureMgmt**][“**Device**”]

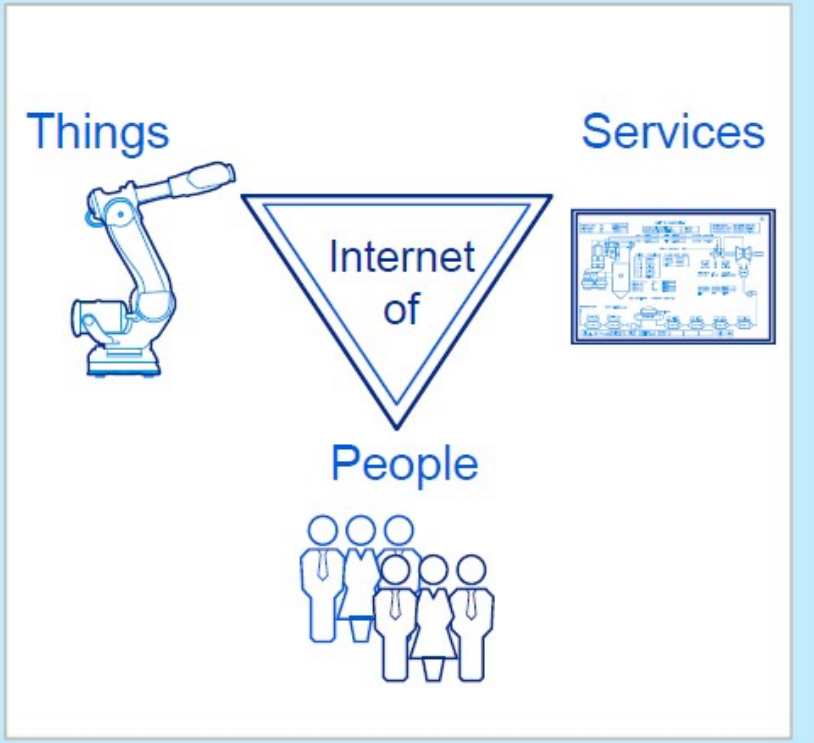
IoTSuite code is available at
https://github.com/pankeshlinux/IoTSuite

SMEWB: Subject Matter Expert Workbench



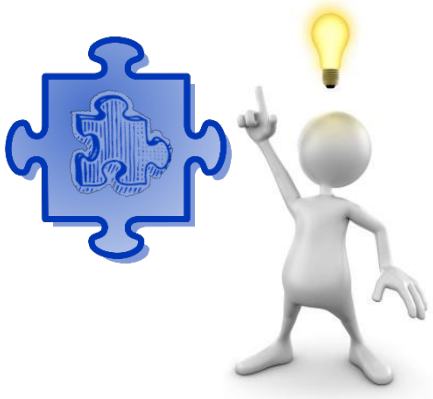
Context





- ABB - industrial sensors, industrial robots, process control systems, etc.
- Blending business solutions with industrial analytics that incorporate deep knowledge of ABB's technical SMEs on equipment and verticals

SMEWB - Key goals



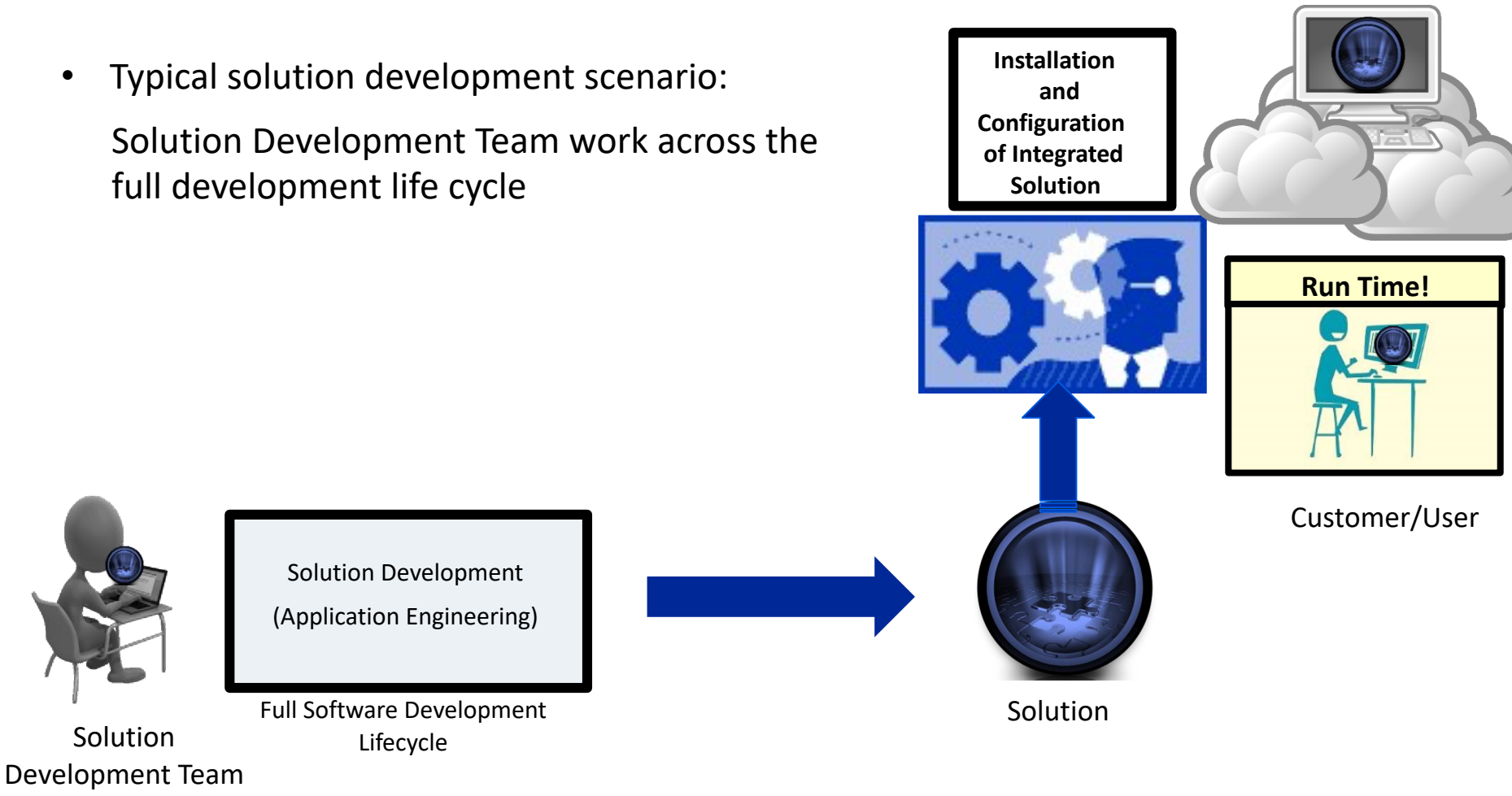
- Accelerate creation, reuse, evolution and delivery of analytic module plugins
  - Minimize SME (**S**ubject **M**atter **E**xpert) effort to share, evolve and reuse knowledge

- Accelerate **integration** of analytics into ABB systems and solutions
  - Minimize business value of knowledge for ABB customers

The old lifecycle: solution development

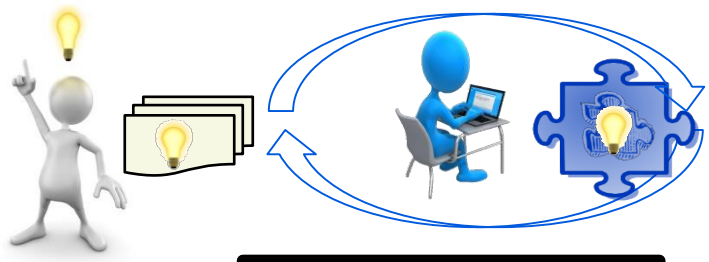
- Typical solution development scenario:

  Solution Development Team work across the full development life cycle



The old lifecycle: solution & analytic development

Blending business solutions and analytics, SMEs write application logic in a flowchart diagram and hire a developer. The developer iterate with SMEs to get the logic right.



Subject Matter Experts (SMEs)

Analytic Module Development

Solution Development Team

Solution Development (Application Engineering)

Full Software Development Lifecycle

Solution

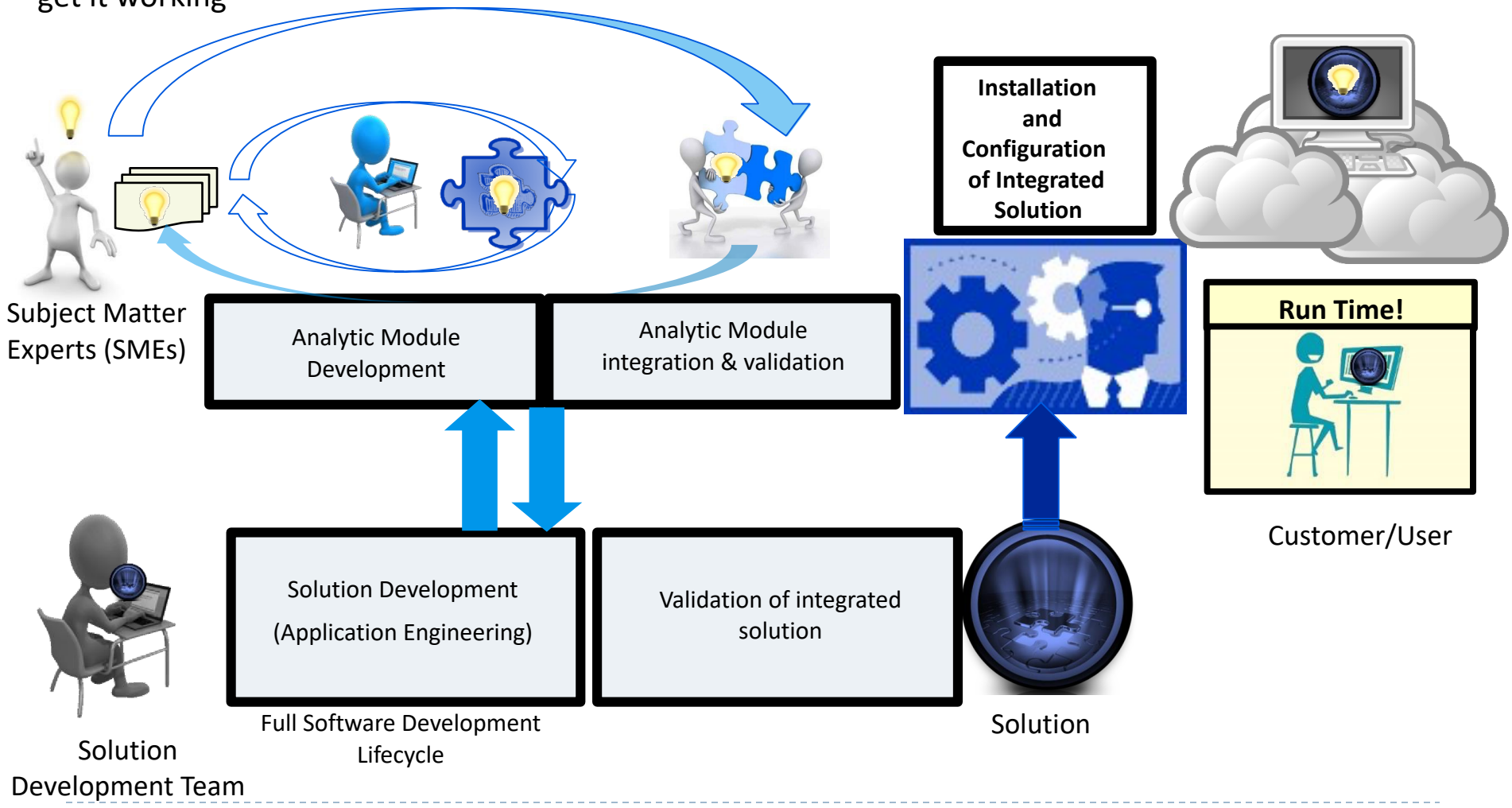
Installation and Configuration of Integrated Solution

Run Time!

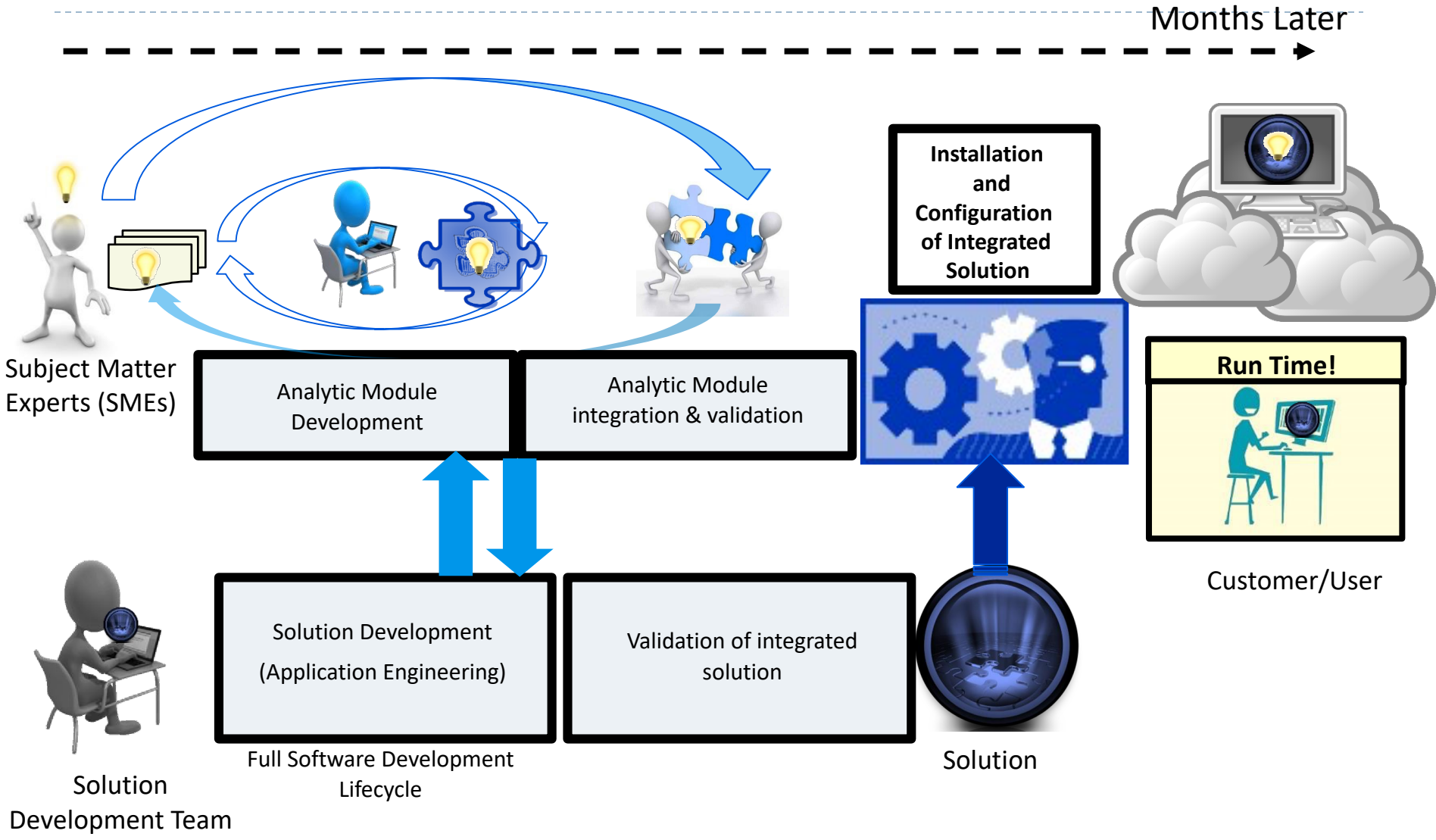
Customer/User

The old lifecycle: integrating analytics into solutions

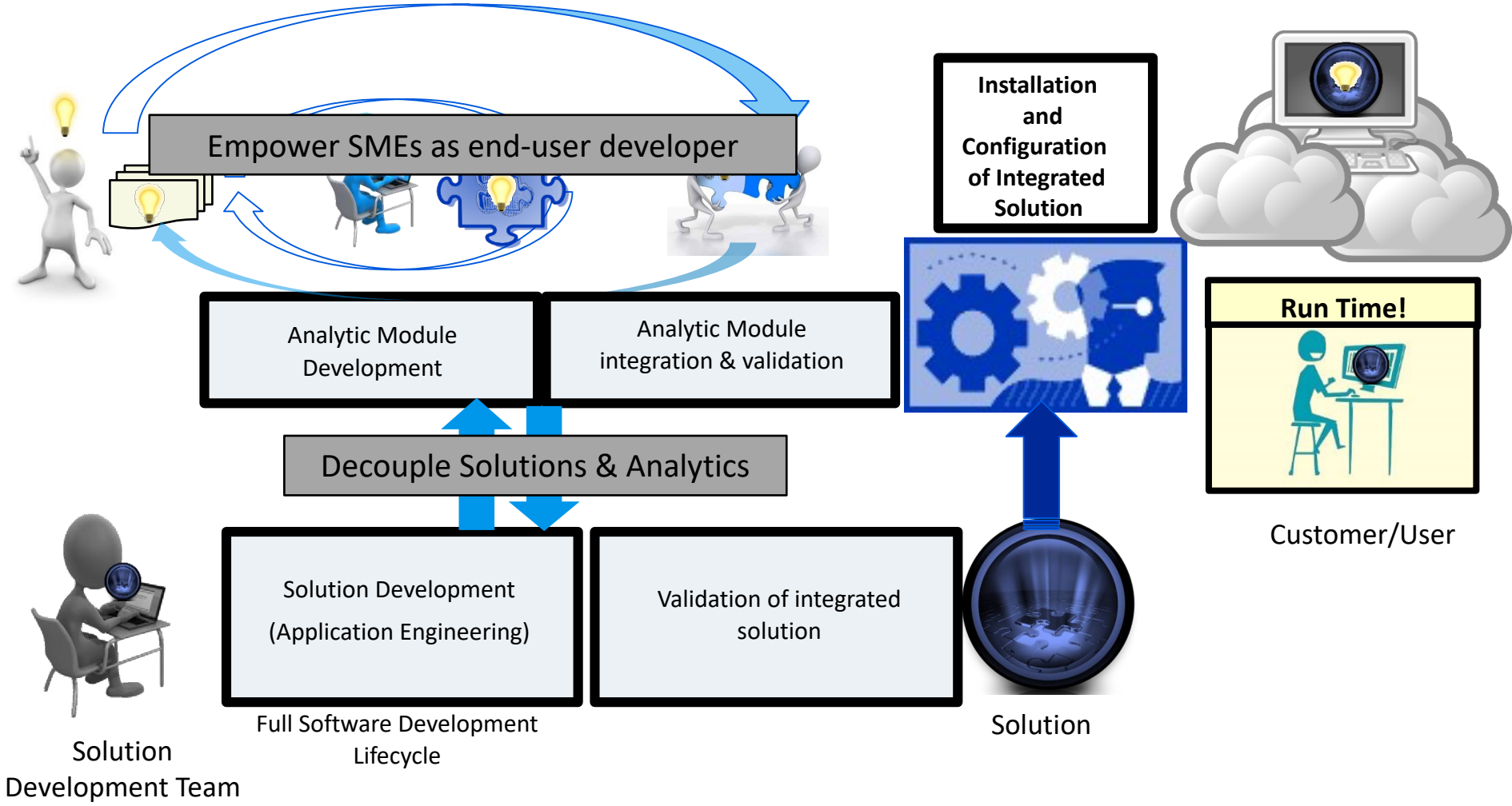
To integrate the analytic module into a solution, the developer iterate with the solution team until they get it working



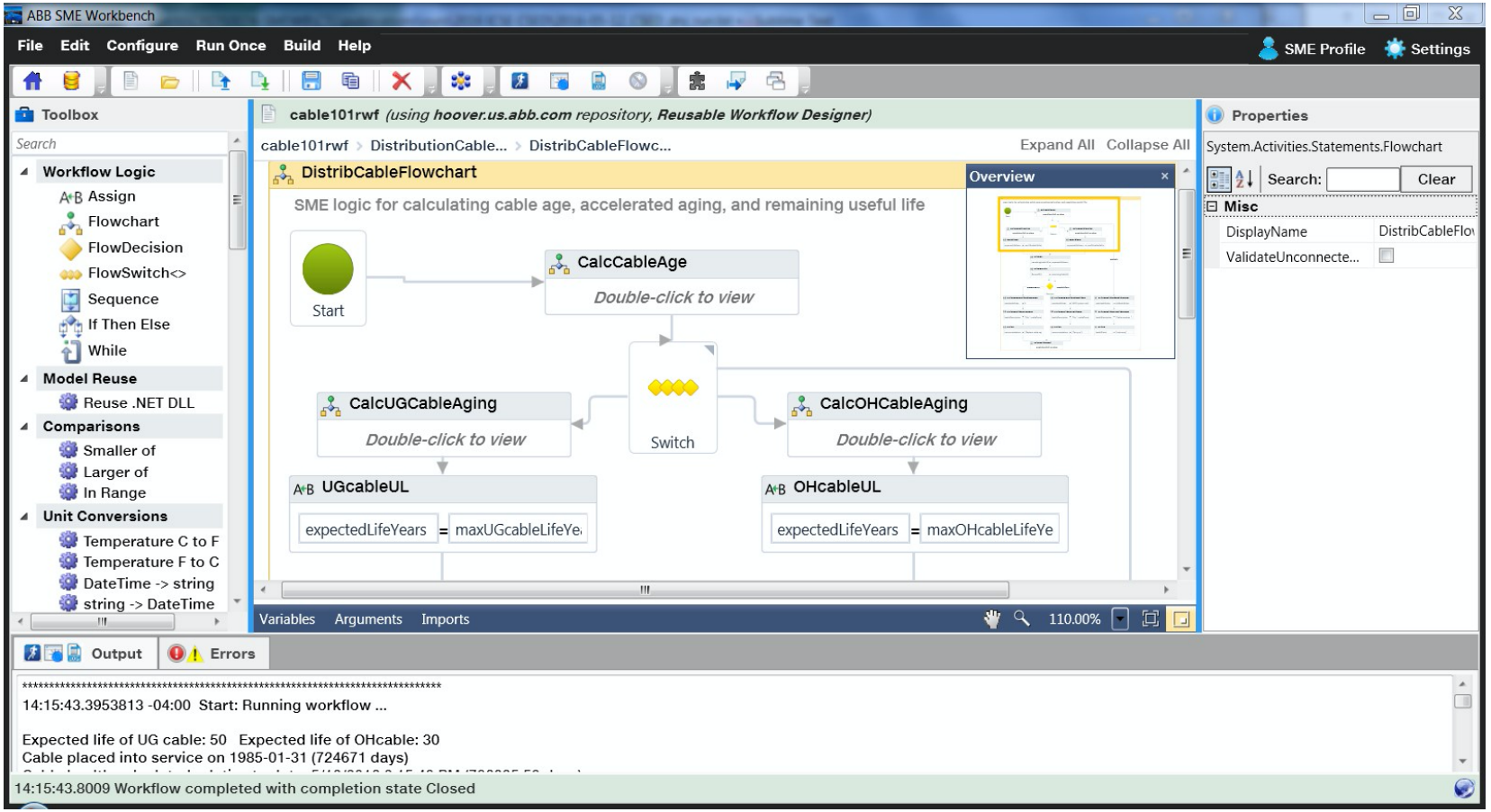
The old lifecycle: integrating analytics into solutions



The old lifecycle: integrating analytics into solutions

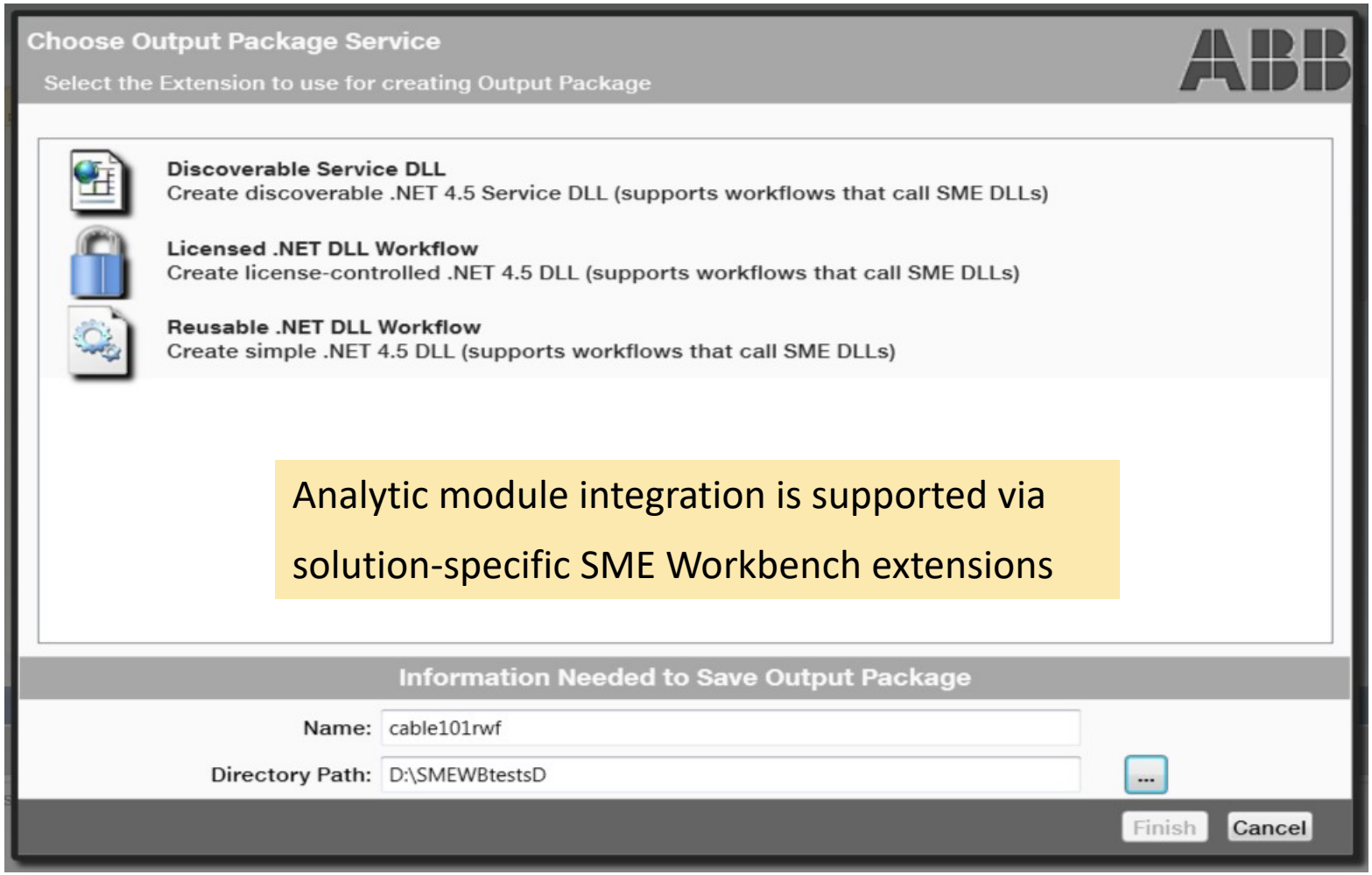


Creating an analytic model using SMEWB



- Drag & Drop to develop analytic modules,
- Reuse existing models from a catalog / MATLAB

Deploying an analytic model



Confirm
Smart Manufacturing

Confirm
Smart Manufacturing

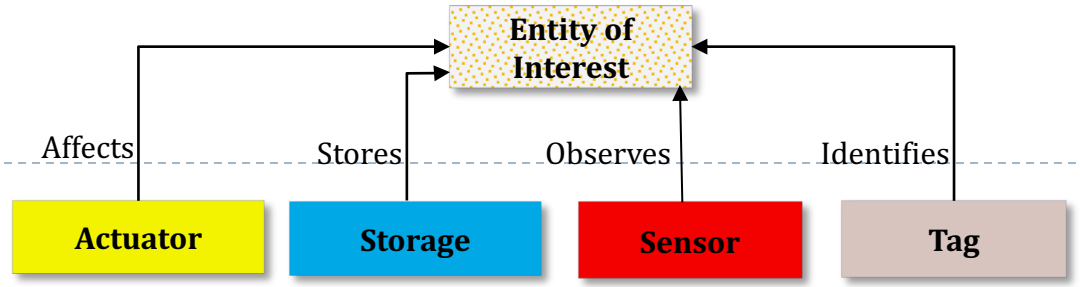
Contact: Pankesh Patel
Email: pankesh.patel@insight-centre.org

www.confirm.ie

Science Foundation Ireland sfi For what's next

Backup Slides

Conceptual Model



Things (or *Entity of Interest*) is an object, including attributes that describe it, and its state that is relevant from a user or an application perspective.

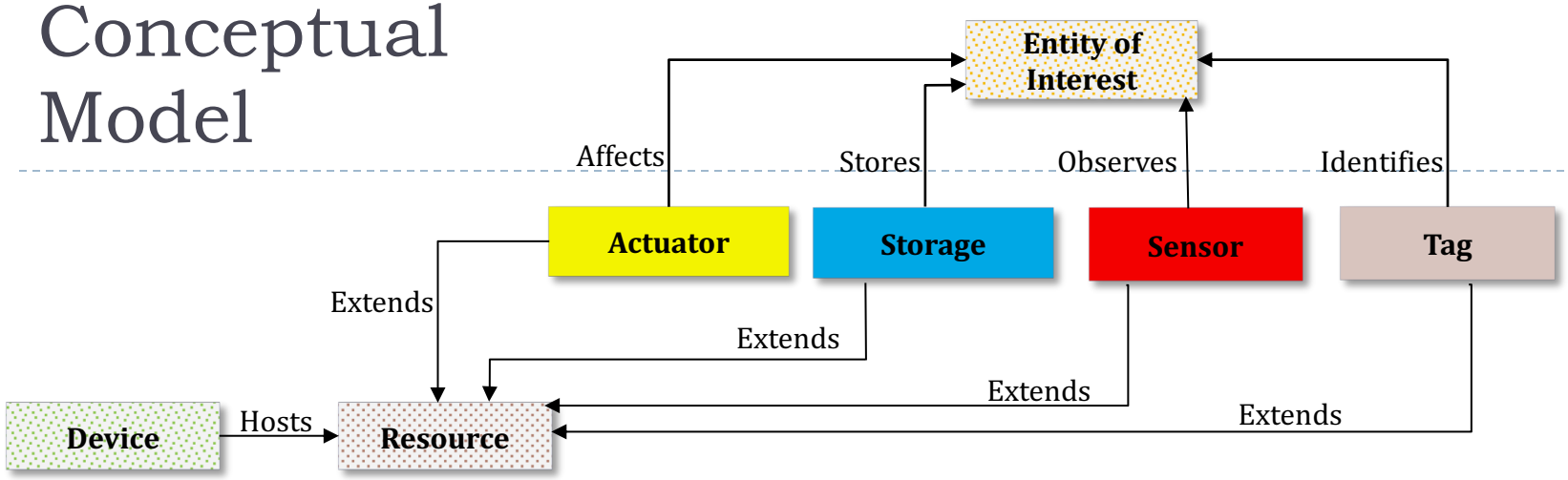
Heart rate

Temperature value of a **Room**

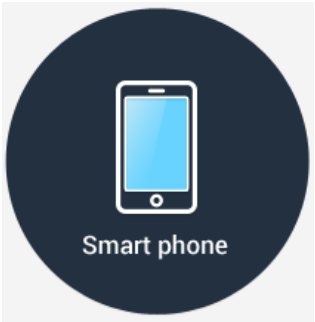
Battery level of a **Smartphone**

- Temperature sensor observes the temperature value of a room
- Heater controls the temperature of a room
- Profile Database stores user's various preferences
- Badge identifies a person.

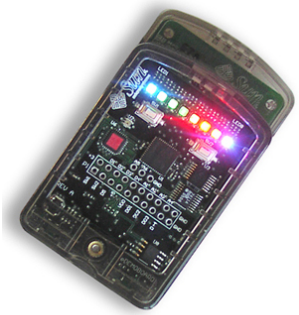
Conceptual Model



- A resource is an abstract representation of a sensor, an actuator, a storage
- A device is an entity that provides resources the ability of interacting with other devices.



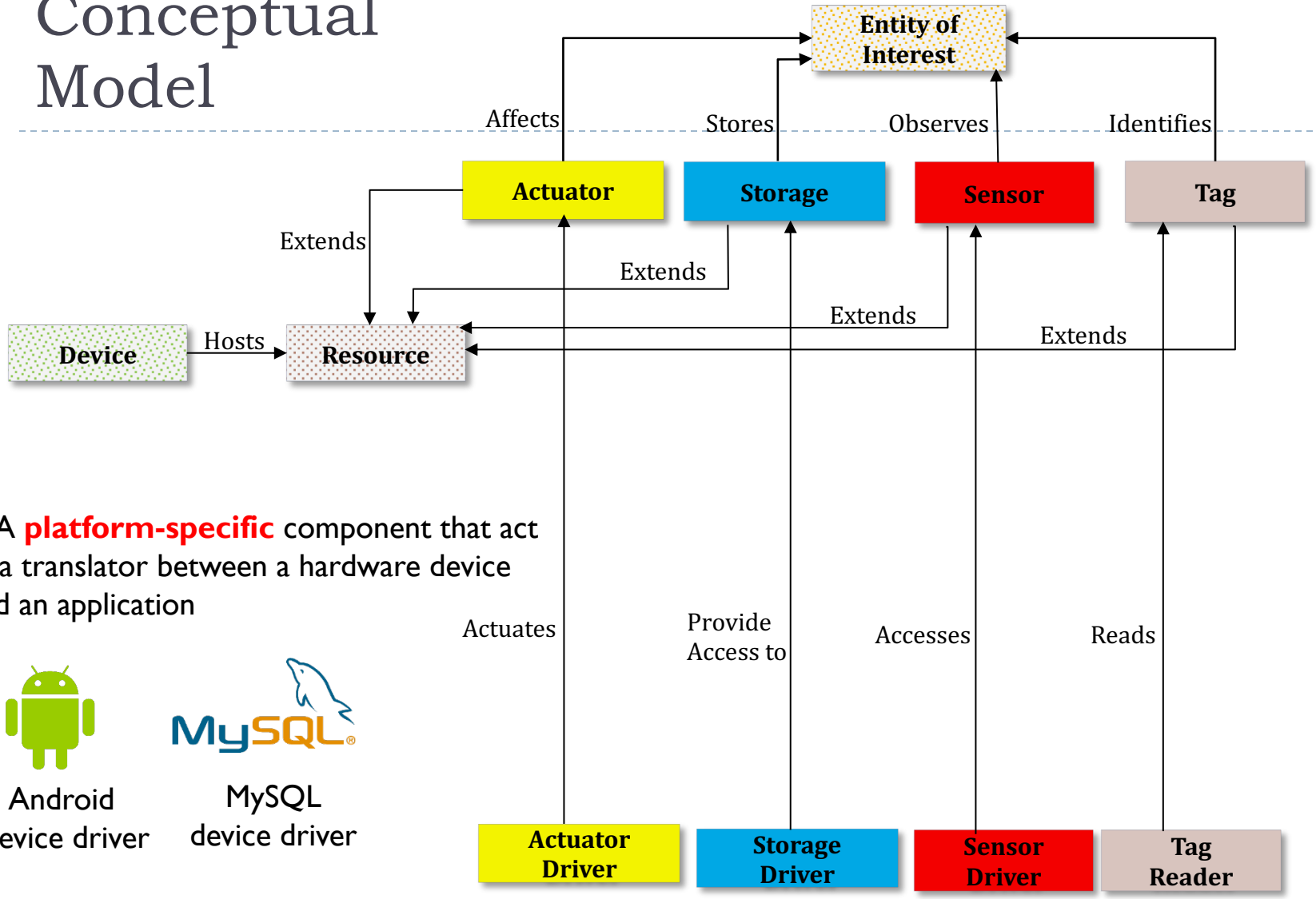
Smart Phones



SunSPOT

Cloud server

Conceptual Model

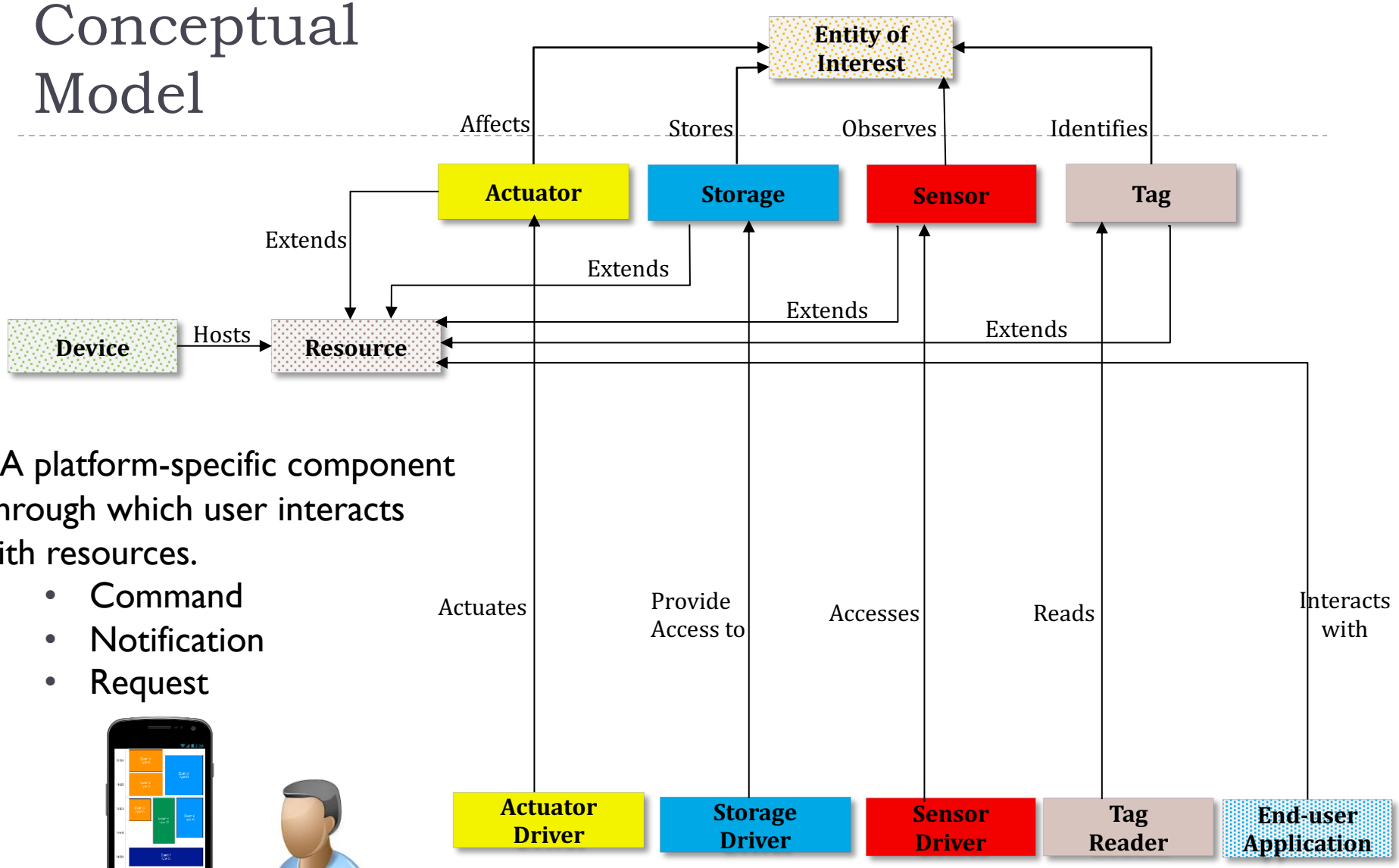


- A **platform-specific** component that act as a translator between a hardware device and an application

Android device driver

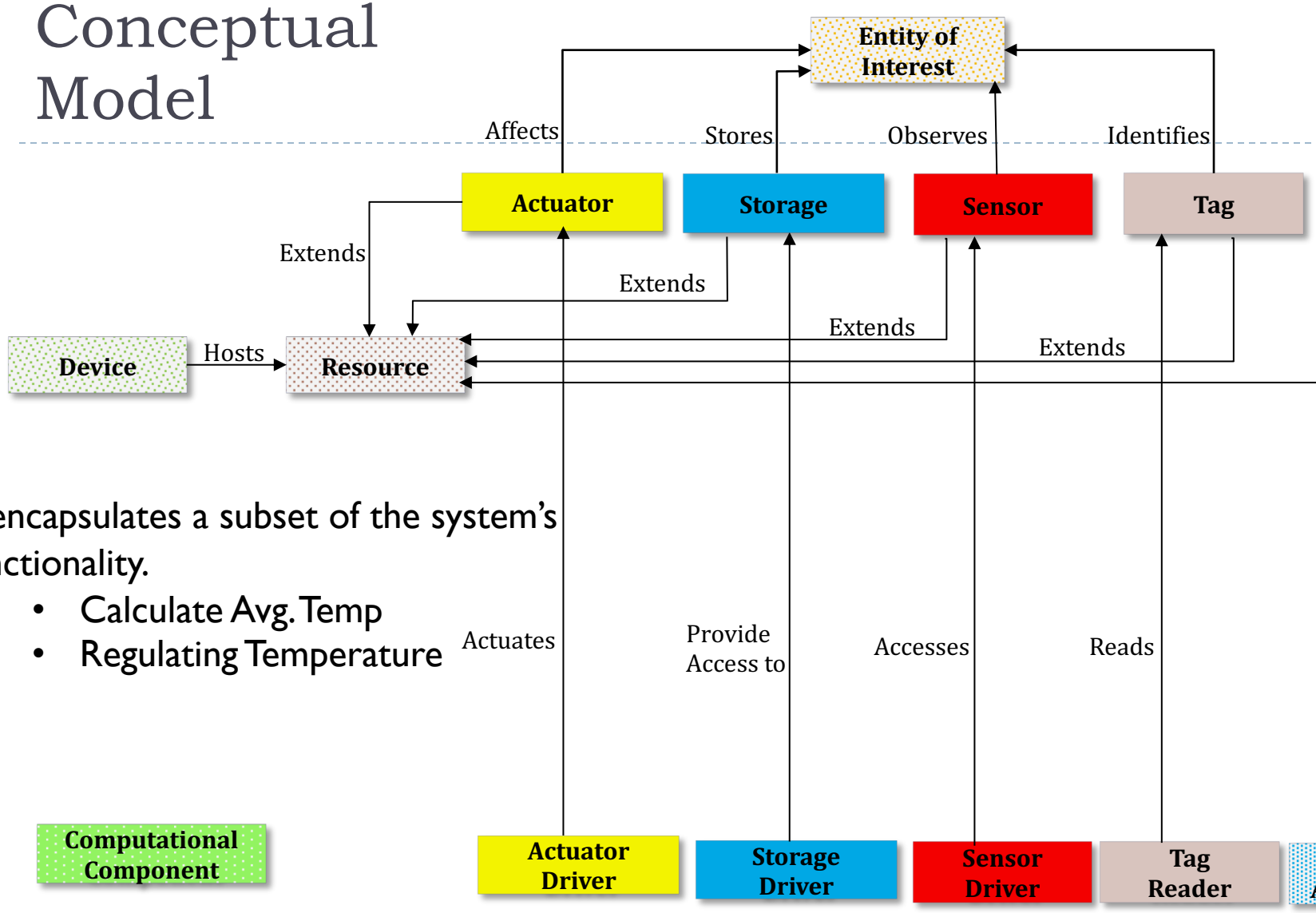
MySQL device driver

Conceptual Model



- A platform-specific component Through which user interacts with resources.
  - Command
  - Notification
  - Request

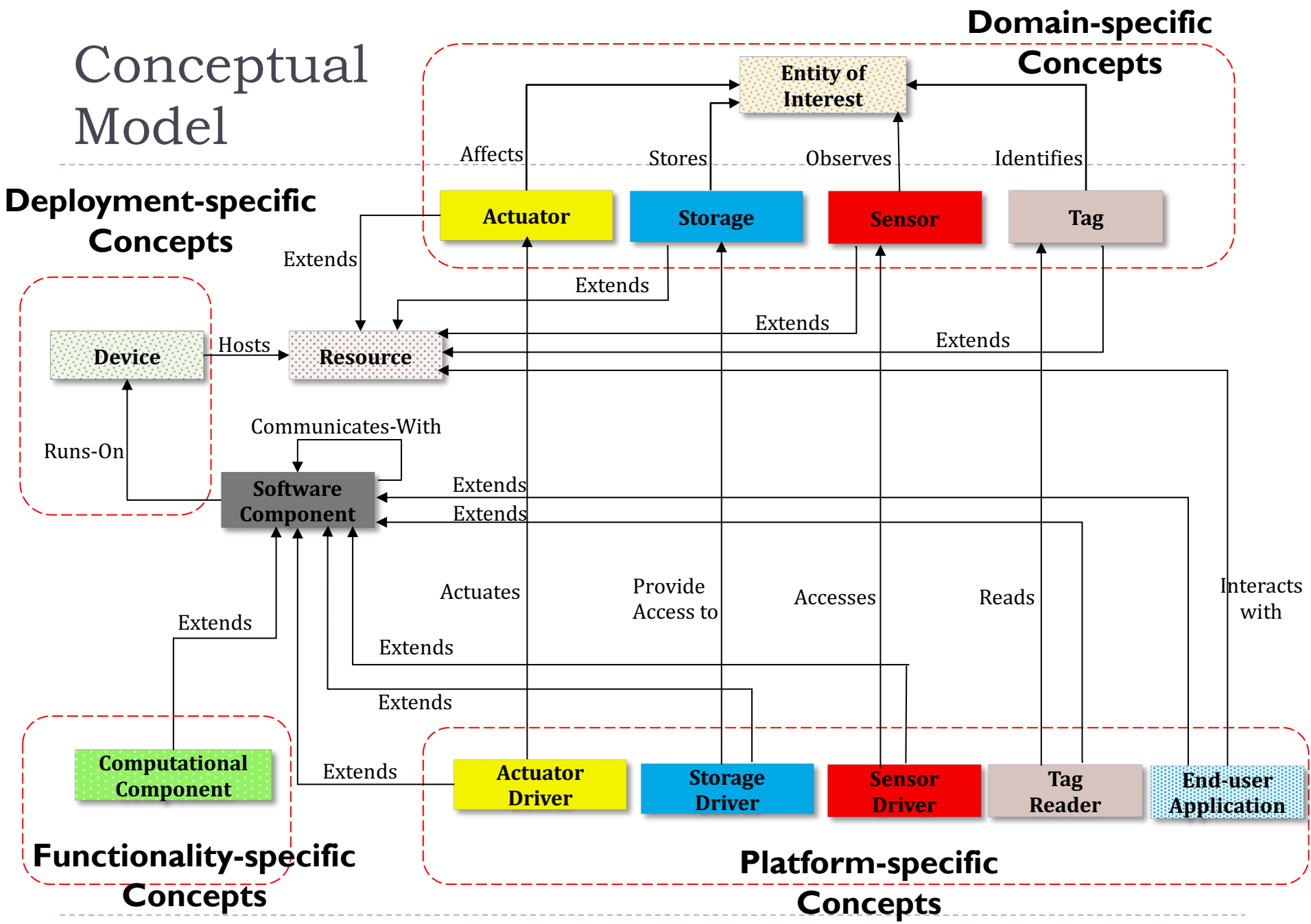
Conceptual Model



It encapsulates a subset of the system's functionality.

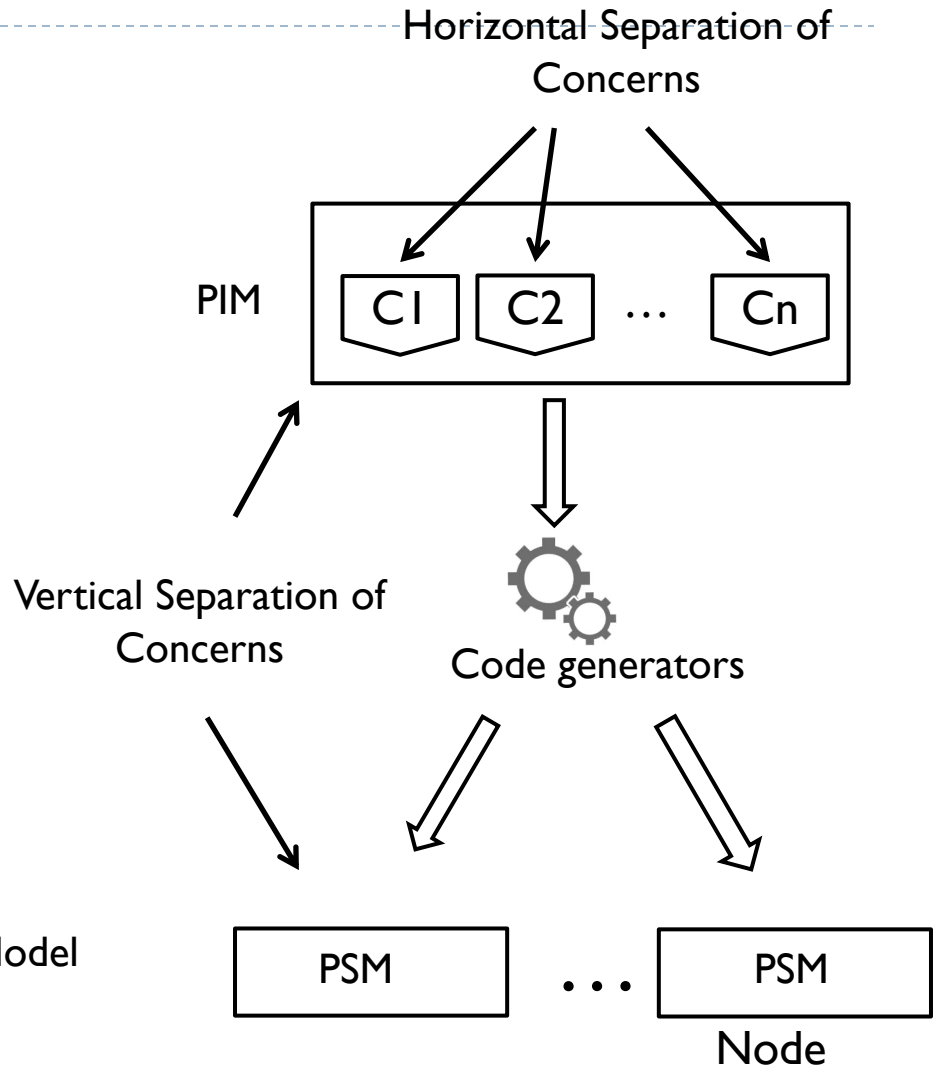
- Calculate Avg. Temp
- Regulating Temperature

Conceptual Model



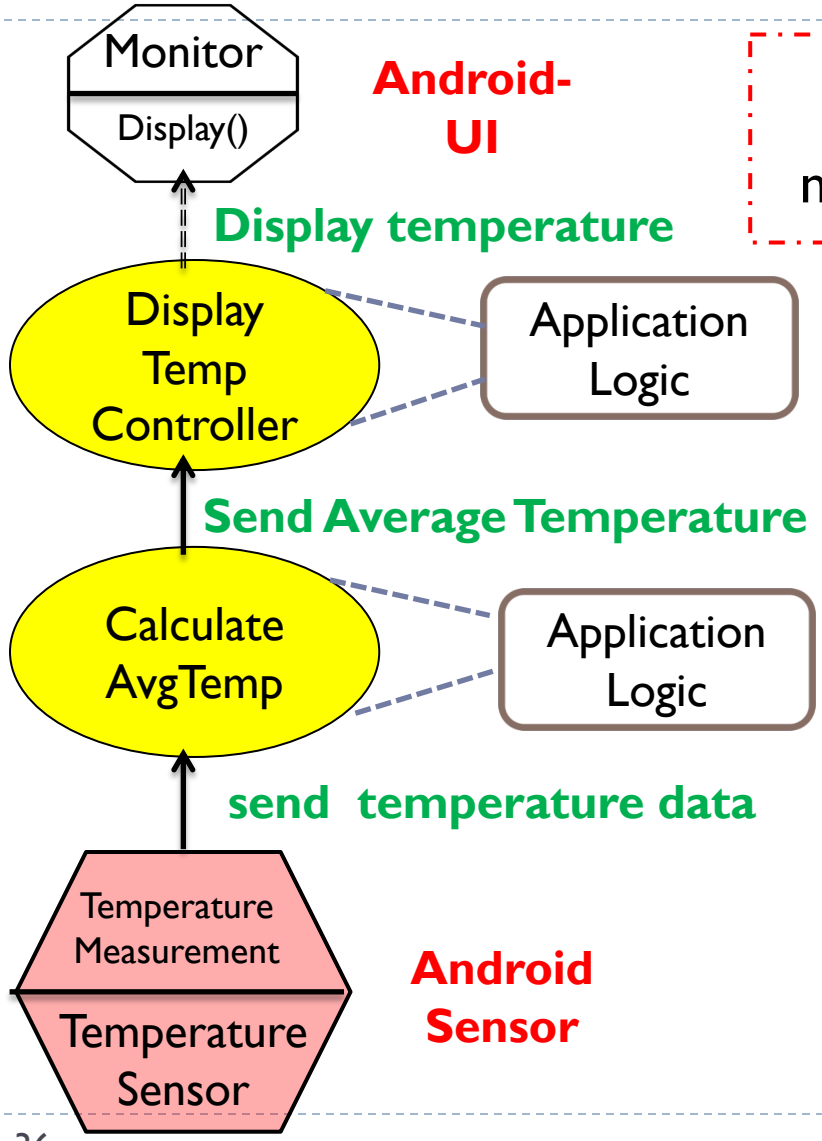
IoTSuite: MDE Approach

- Separation of Concerns (reusability)
- Integration of existing DSL (reduce complexity & effort)
- Automation wherever possible (reduce effort)

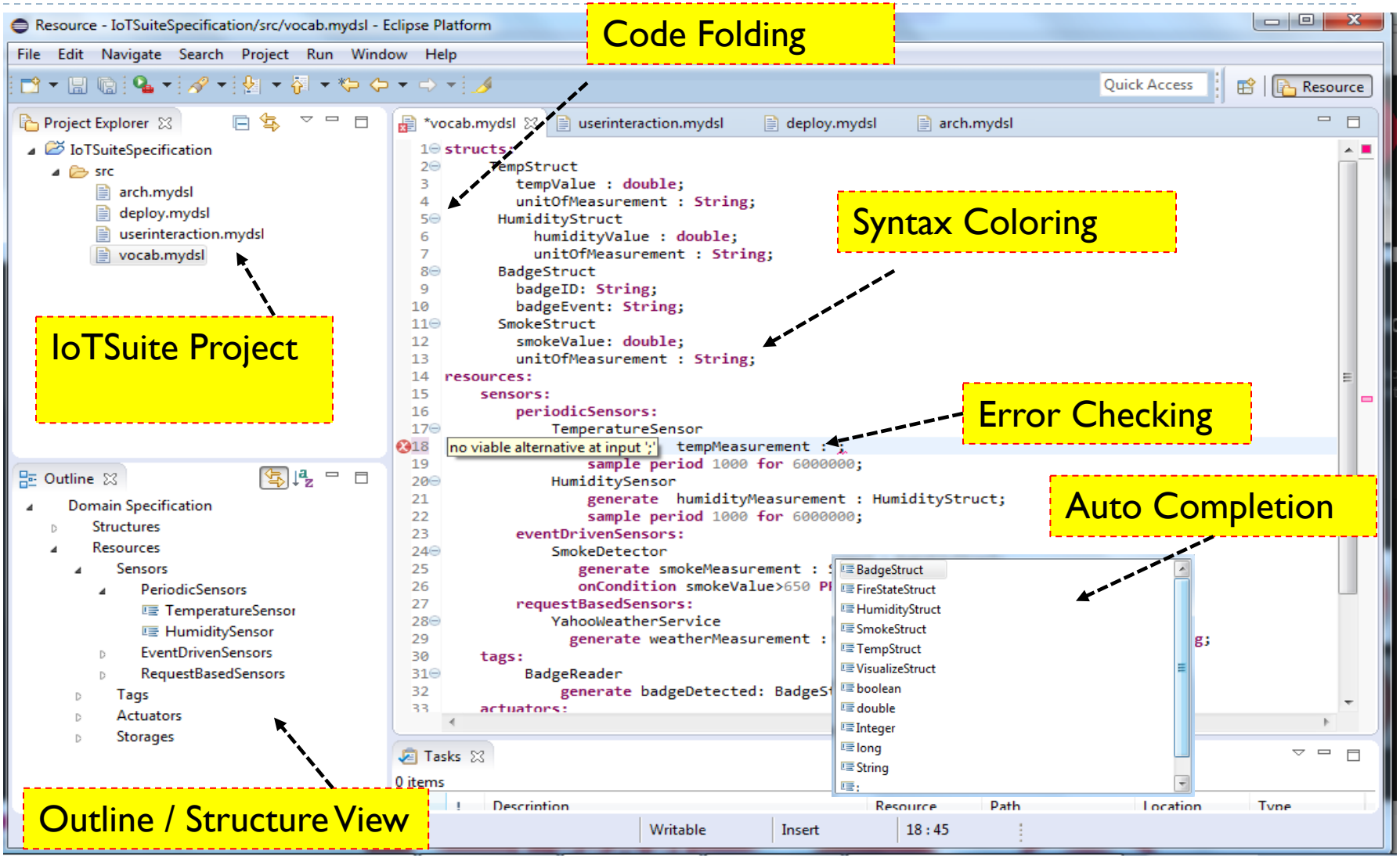


PIM – Platform Independent Model
PSM – Platform Specific Model

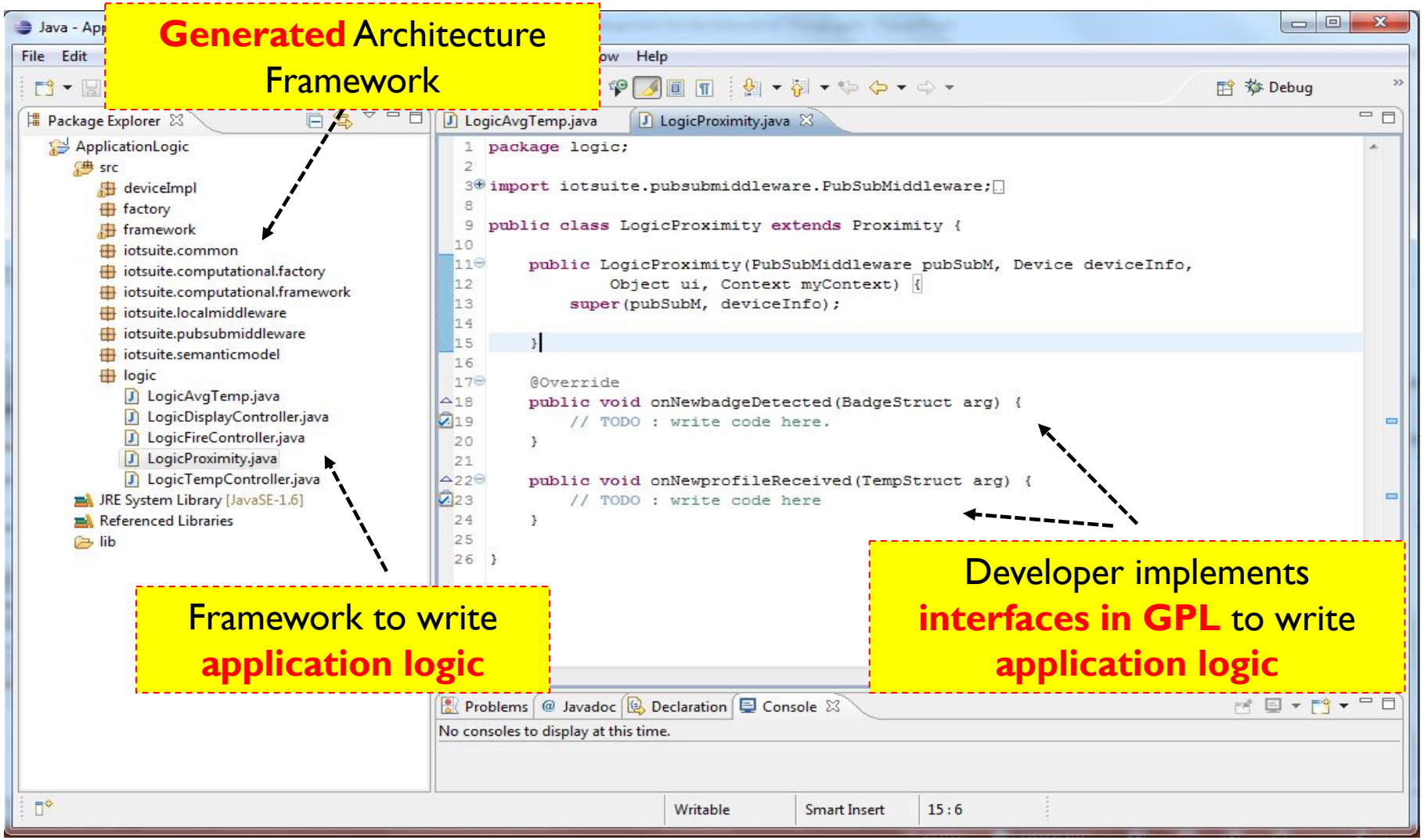
Example


Monitor
Display()
Android-
UI
Notify me temperature value of
my room periodically on my monitor
Display temperature
Display
Temp
Controller
Application
Logic
Send Average Temperature
Calculate
AvgTemp
Application
Logic
send temperature data
Temperature
Measurement
Temperature
Sensor
Android
Sensor

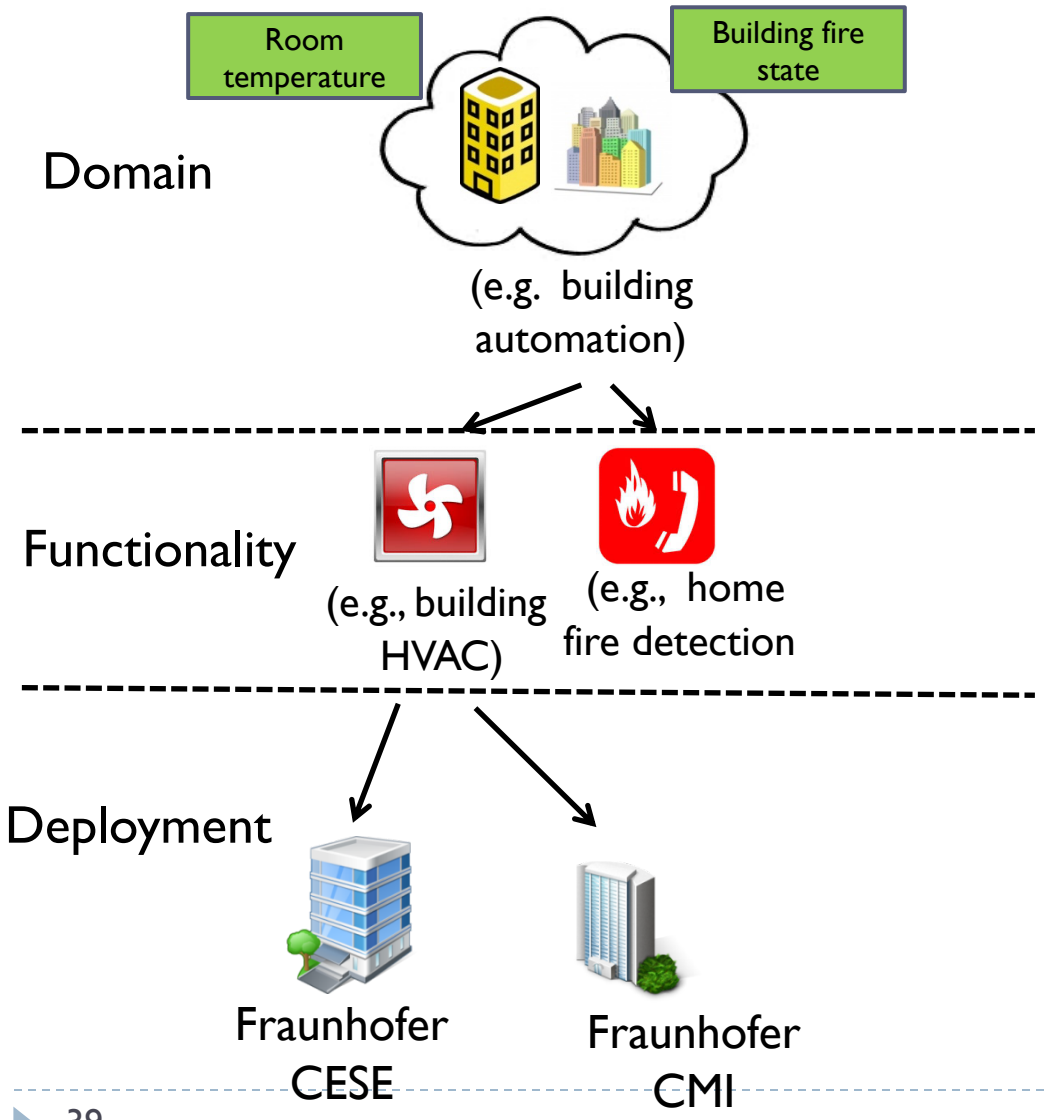
IoTSuite Code editor



IoTSuite architecture framework



Commonality at various levels

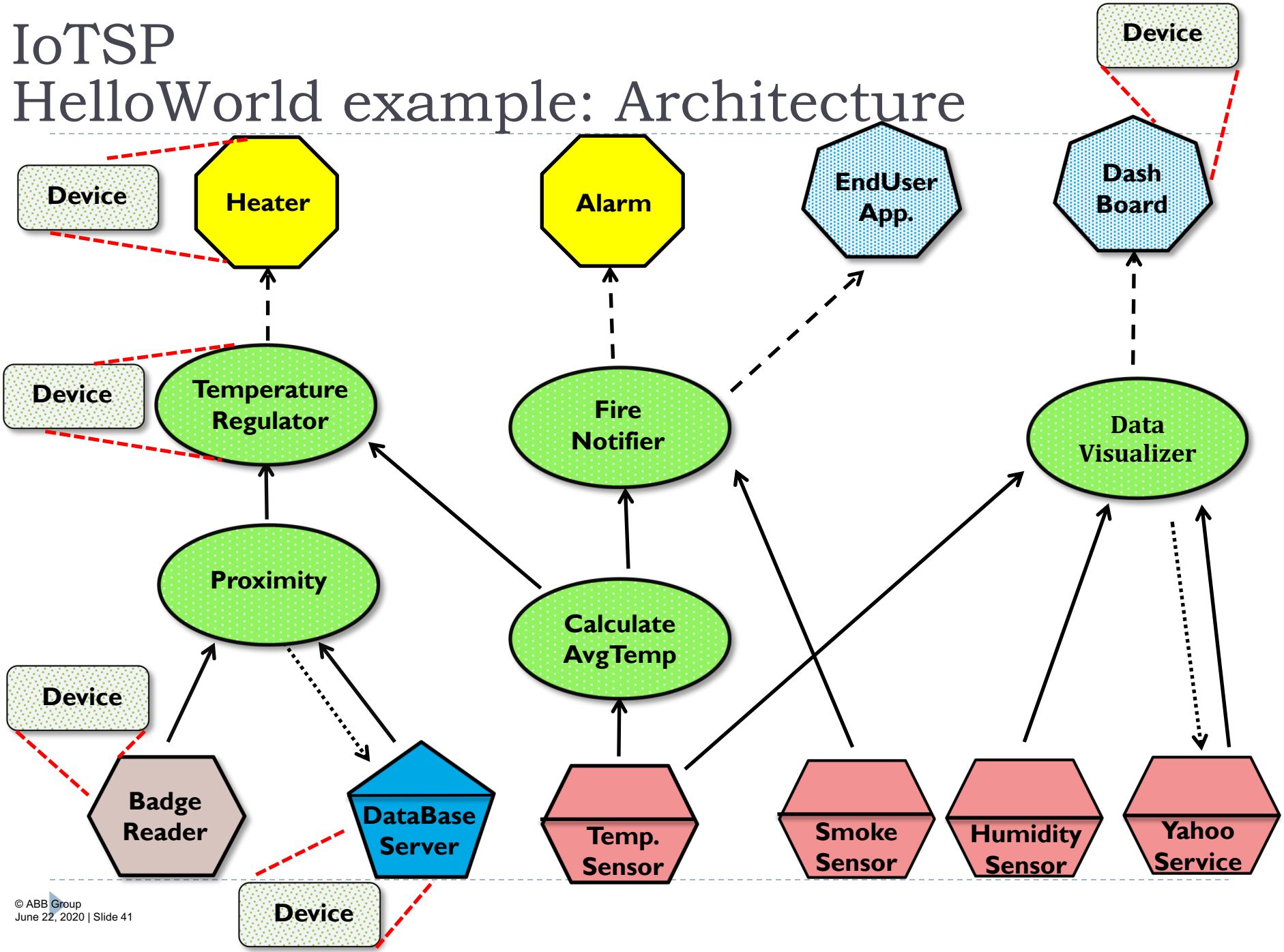


"Reusability across concerns"

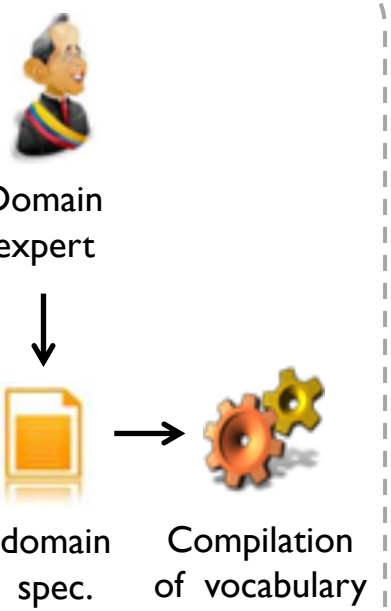
IoTSuite code is available at
https://github.com/pankeshlinux/IoTSuite

The current version of IoTSuite supports several IoT technologies such as Android, Raspberry PI, Arduino, and JavaSE-enabled devices, Messaging protocols such as MQTT, CoAP, websocket, Server technologies such as node.js, Relational database such as MySQL, and Microsoft Azure Cloud services.

IoTSP
HelloWorld example: Architecture



Our approach: Domain concern

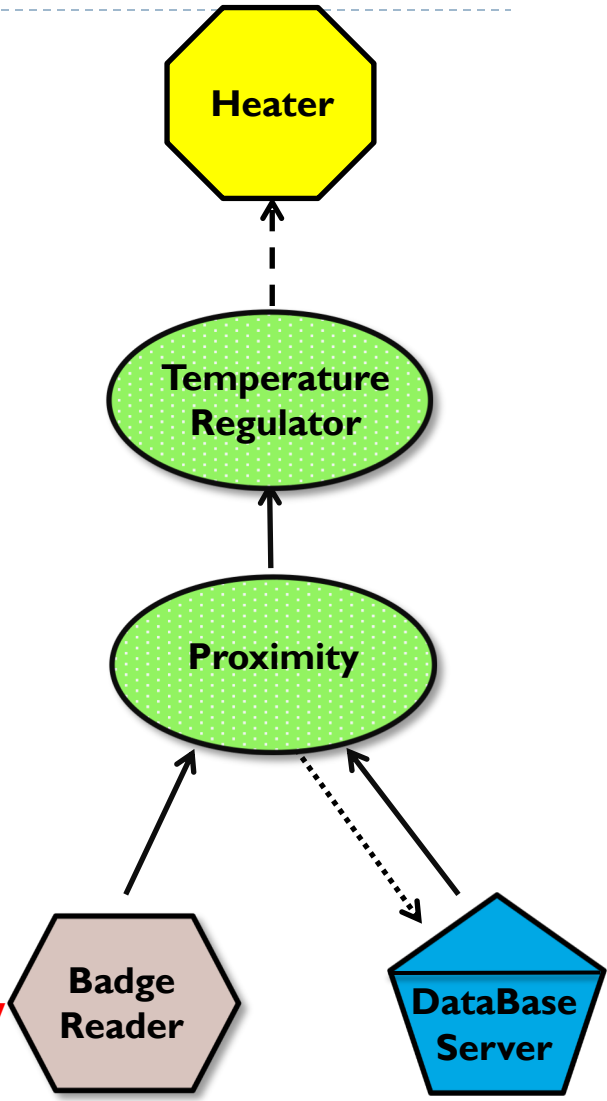


This step involves the specification of concepts that are responsible for interacting with Entities of Interests (EoI).

- Sensors (Sense the EoI)
- Actuators (Affect the EoI)
- Storage (Store information about the EoI)
- Tags (Identify the EoI)

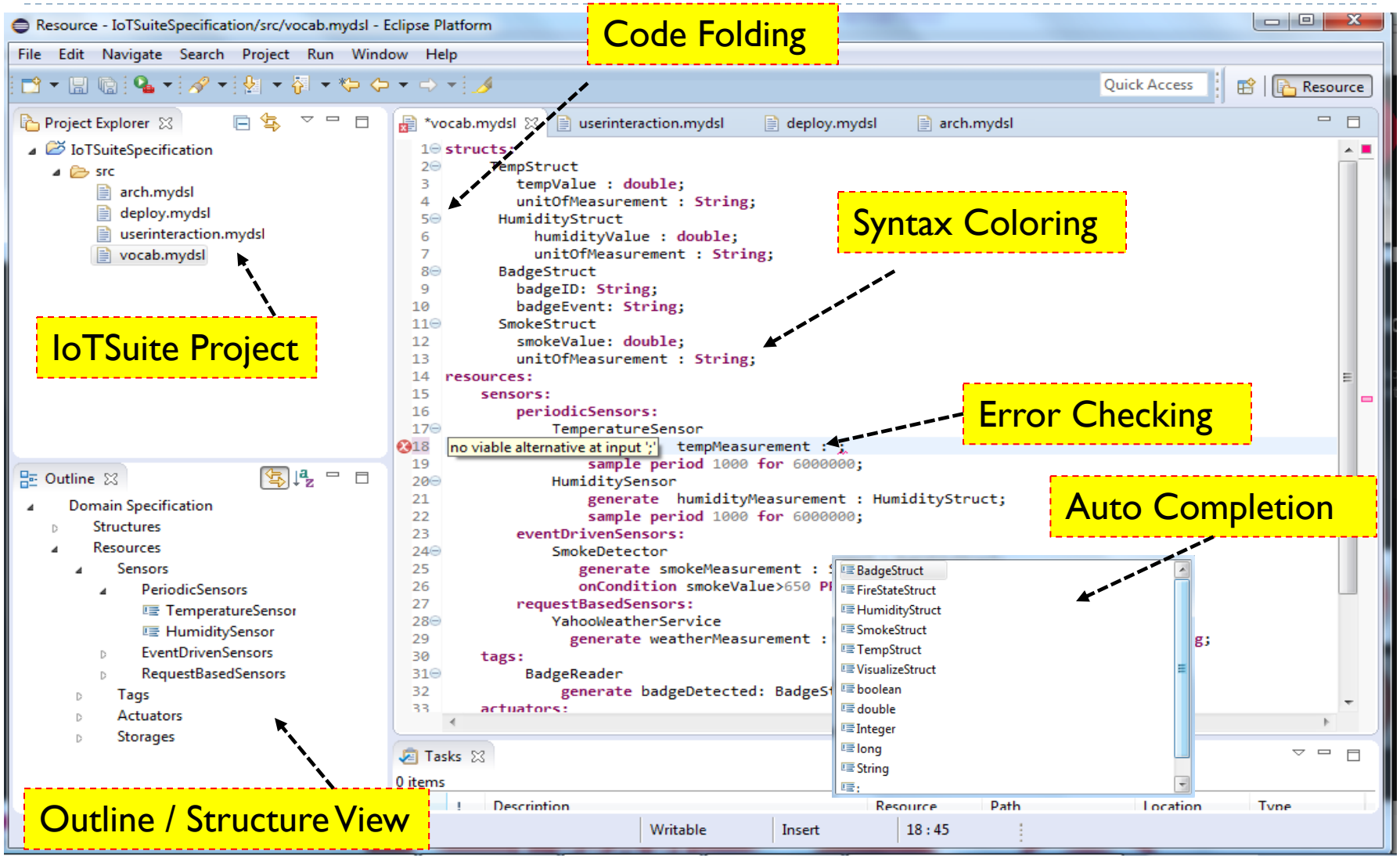
IoTSP
Domain language: code snippet

```
structs:
    BadgeStruct
        badgeID: String;
        badgeEvent:String;
    TempStruct
        tempValue: double;
        unitOfMeasurement: String;
resources:
    Sensors:
        BadgeReader
            generate badgeDetected: BadgeStruct;
    actuators:
        Heater
            action Off();
            action SetTemp(setTemp: TempStruct);
    storages:
        ProfileDB
        generate  profile: tempStruct accessed-by
                  badgeID: String;
```

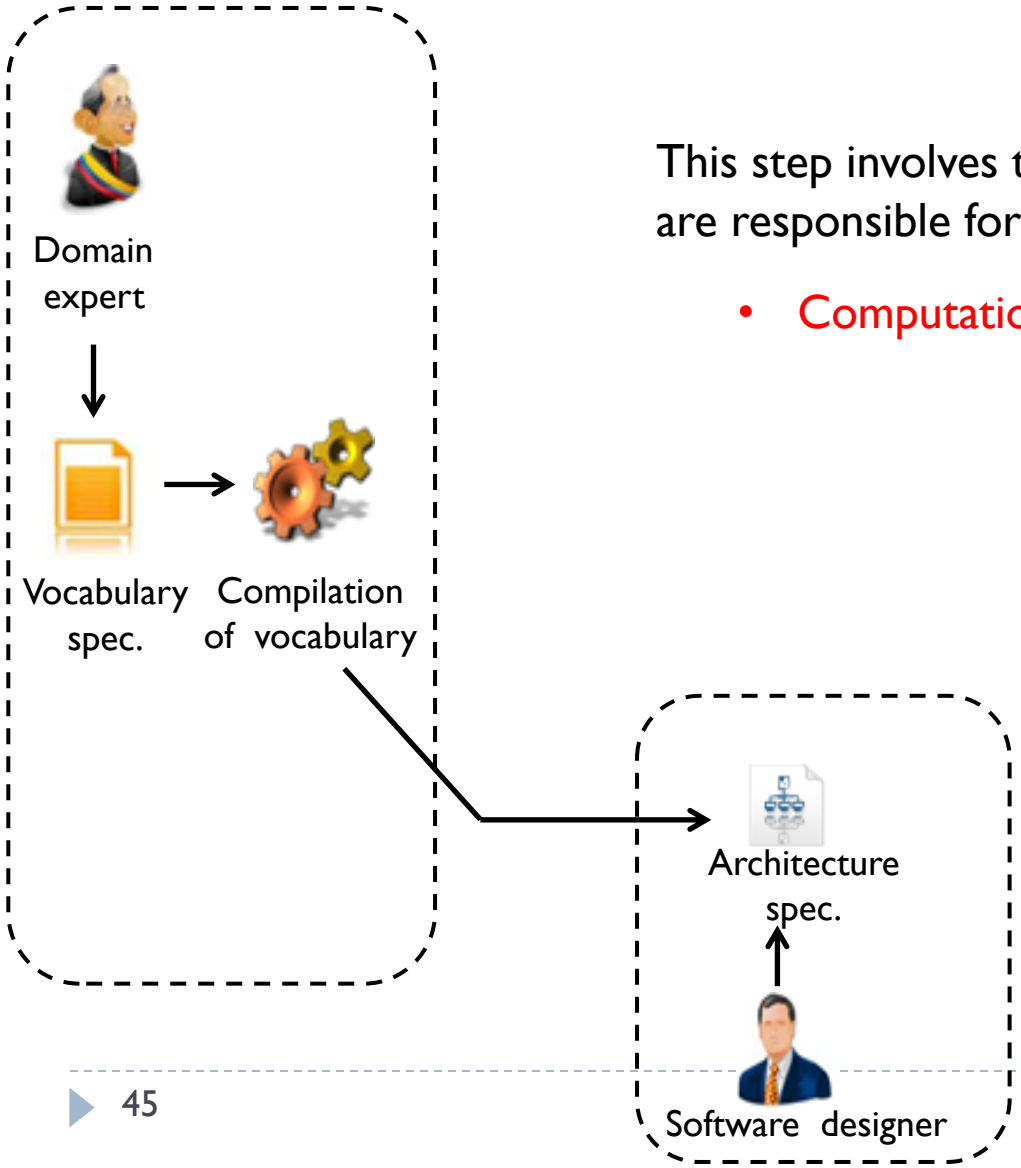


- Abstractions to specify heterogeneous entities
- One entity description for many instances

IoTSP
IoTSuite Code Editor



Our approach: functional concern



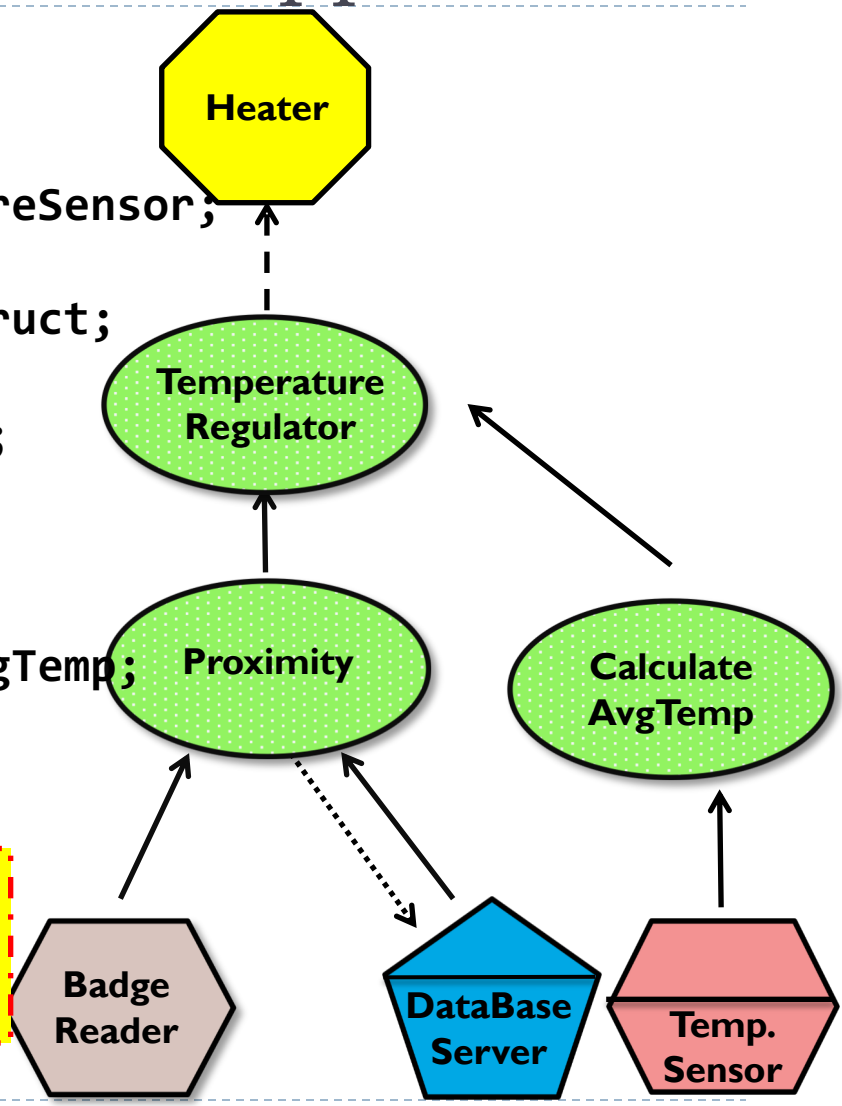
This step involves the specification of concepts that are responsible for computation or processing.

- Computational Service

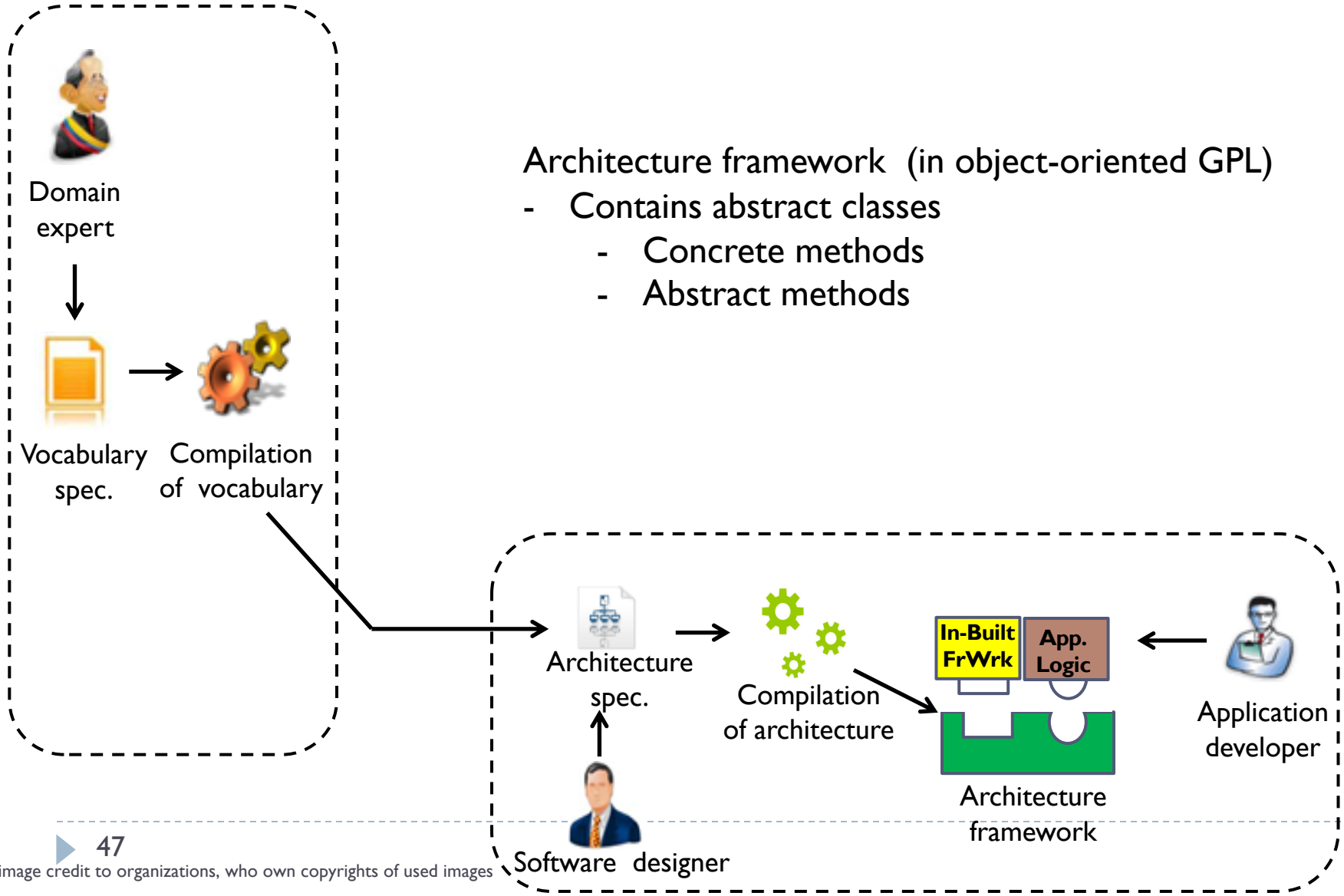
IoTSP
Architecture language: code snippet

```
computationalService:
 AvgTemp
  consume tempMeasurement from TemperatureSensor;
  COMPUTE (AVG_BY_SAMPLE,5);
  generate roomAvgTempMeasurement:TempStruct;
 Proximity
  consume badgeDetected from BadgeReader;
  request profile to ProfileDB;
  generate tempPref: TempStruct;
 TempController
  consume roomAvgTempMeasurement from AvgTemp;
  consume tempPref from Proximity;
  command SetTemp(setTemp) to Heater;
```

- **Programming constructs to hide heterogeneous interactions**
- **Programming constructs to perform common operations.**



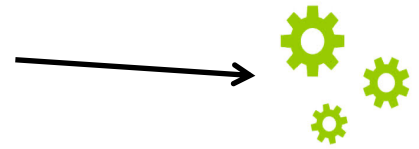
Our approach: functional concern



IoTSP
Architecture framework: code snippet

Structured code, Application Developer has to only implement abstract methods.

```
Proximity
   consume badgeDetected from BadgeReader;
```



Compiler

Concrete method for Subscription request

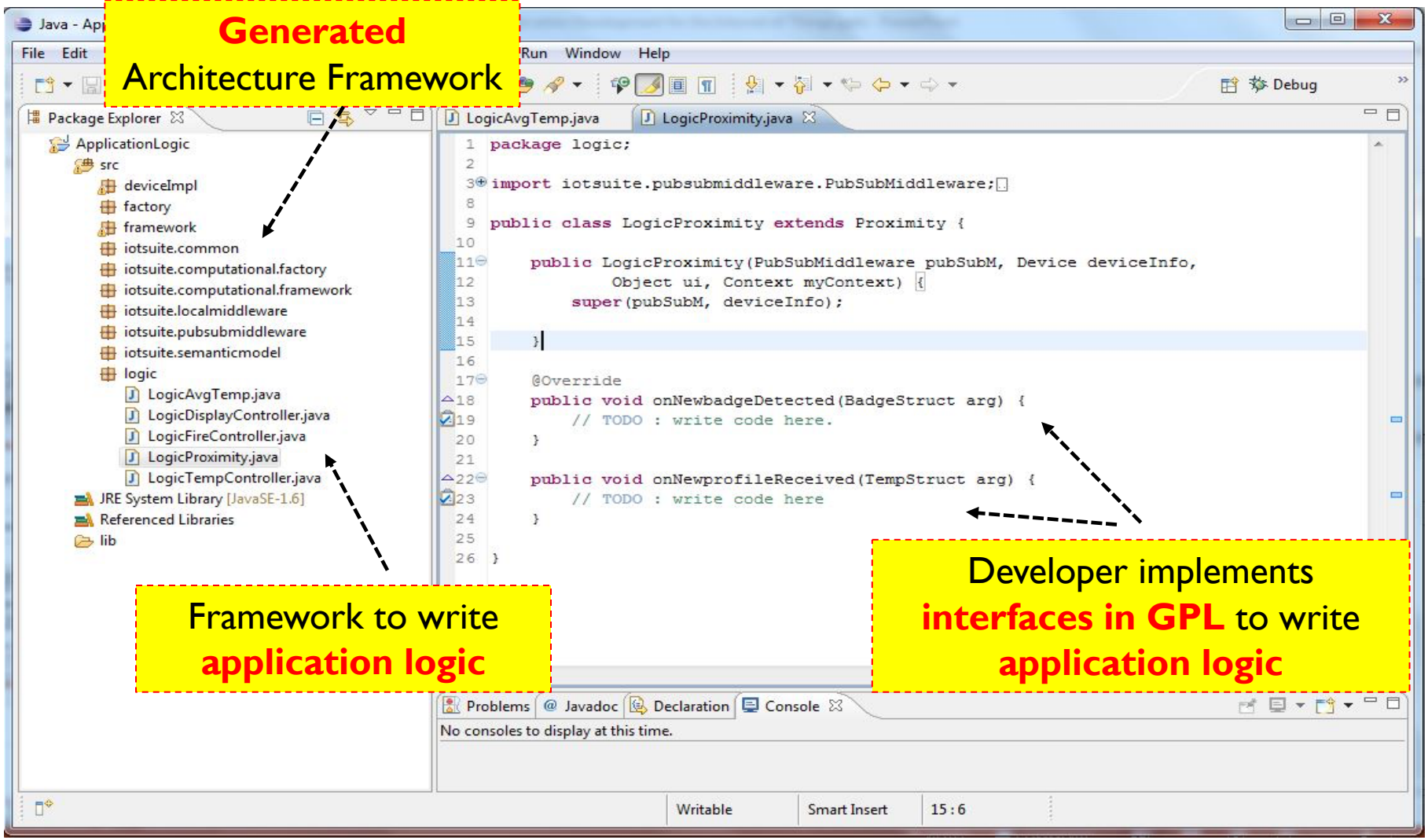
Concrete method for Receiving notification

```
public void subscribeBadgeDetected() {
        PubSubMiddleware.subscribe(this, “badgeDetected",
subscriptionCondition);
}
public void notifiedReceived (String event Name, Object
arg, Device  deviceInfo) {
      if (eventName.equals(“badgeDetected”) {
           onBadgeDetectedEvent((BadgeStruct) arg) ;
    }
}
```

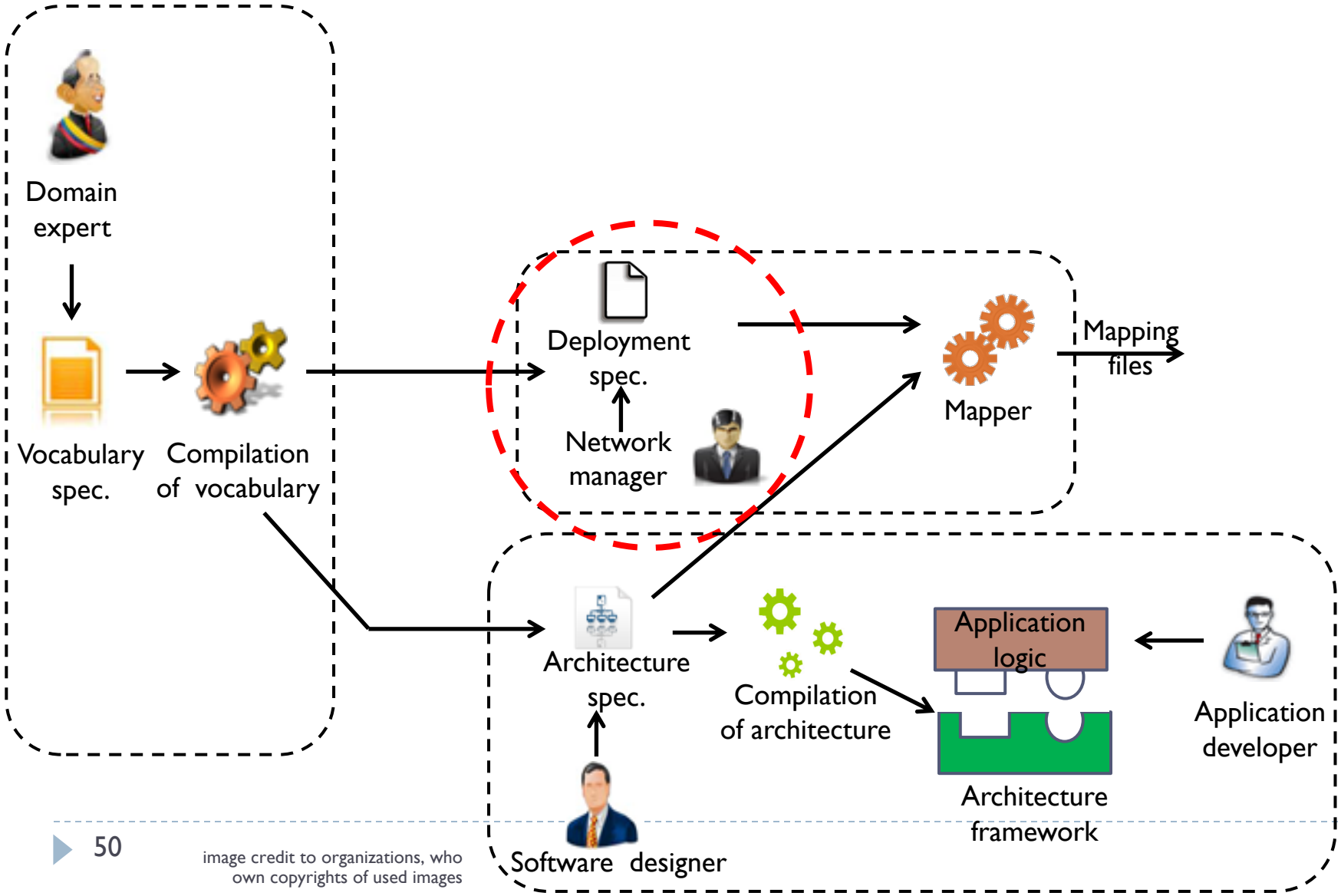
Application developer

```
public abstract void onBadgeDetectedEvent(BadgeStruct arg);
```

IoTSP
IoTSuite architecture framework



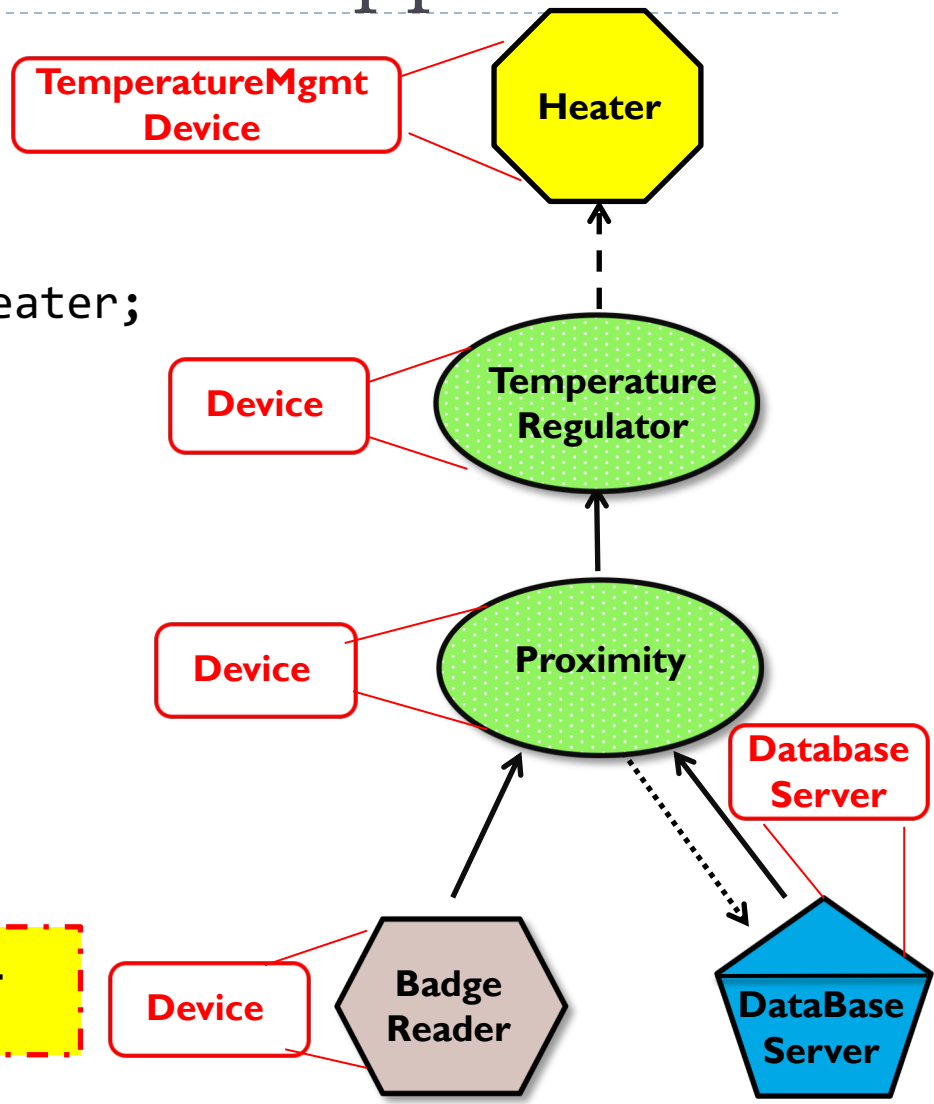
Deployment concern



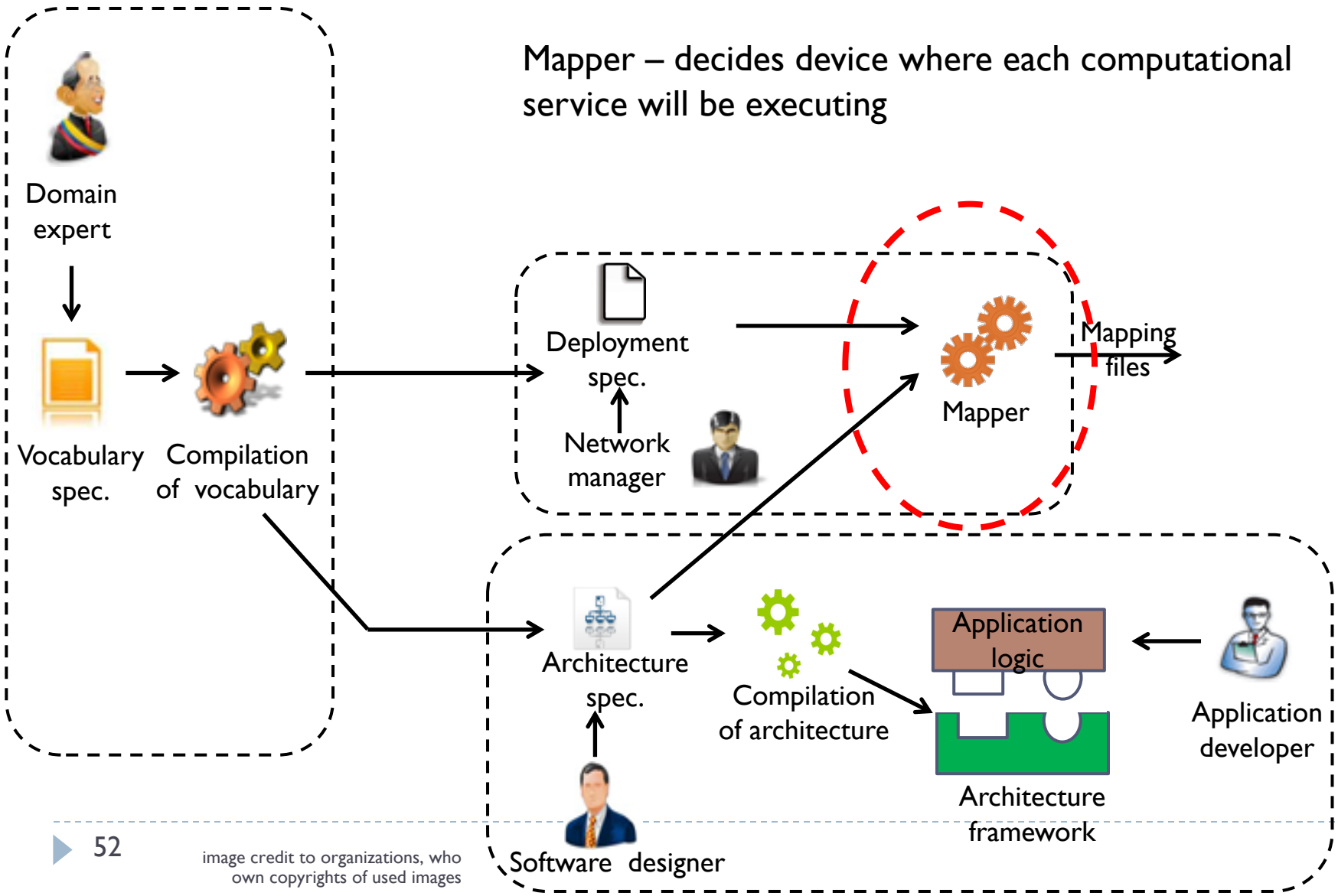
IoTSP
Deployment language: code snippet

```
TemperatureMgmtDevice:
    location:
        Room:1;
    platform: NodeJS;
    resources: TemperatureSensor, Heater;
    protocol: mqtt;
DatabaseServer:
    location:
        Room:1;
    platform: JavaSE;
    resources: ProfileDB;
    protocol: mqtt;
    database: MySQL;
...
```

Property of each device is specified individually – Not Scalable



Deployment concern



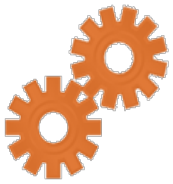
Deployment concern : Mapping

```
computationalService:
    Proximity
      …
    TemperatureController
      …
```

Architecture specification

```
devices:
   Device1:
    …
   Device2:
    …

   DeviceN:
      …
```

Deployment Specification



Mapper

```
Device1:
    Proximity
```

Mapping decision (output in GPL)

```
Device2:
TemperatureController
```

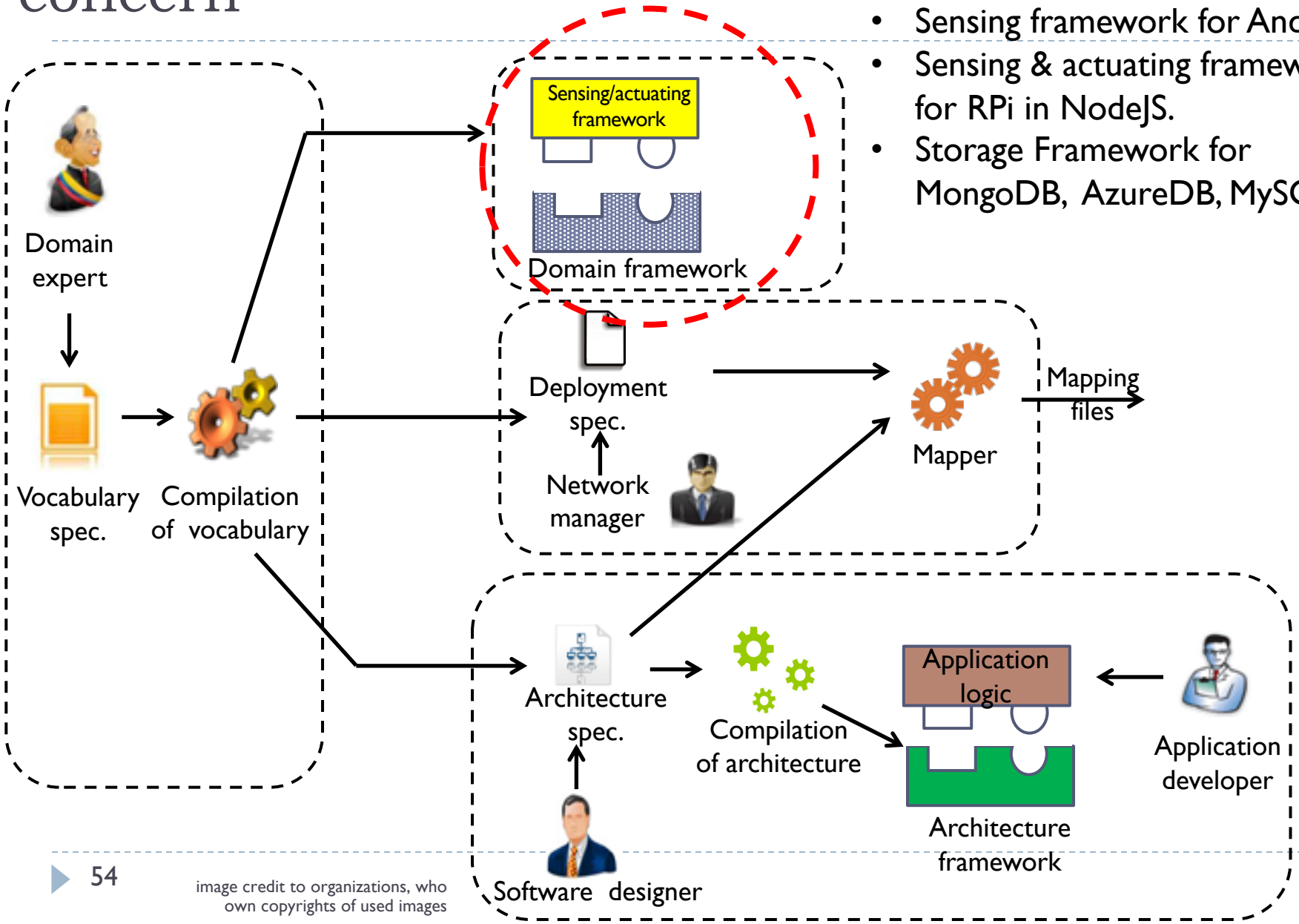
Mapping decision (output in GPL)

Mapping considering device properties is a part of our future work.

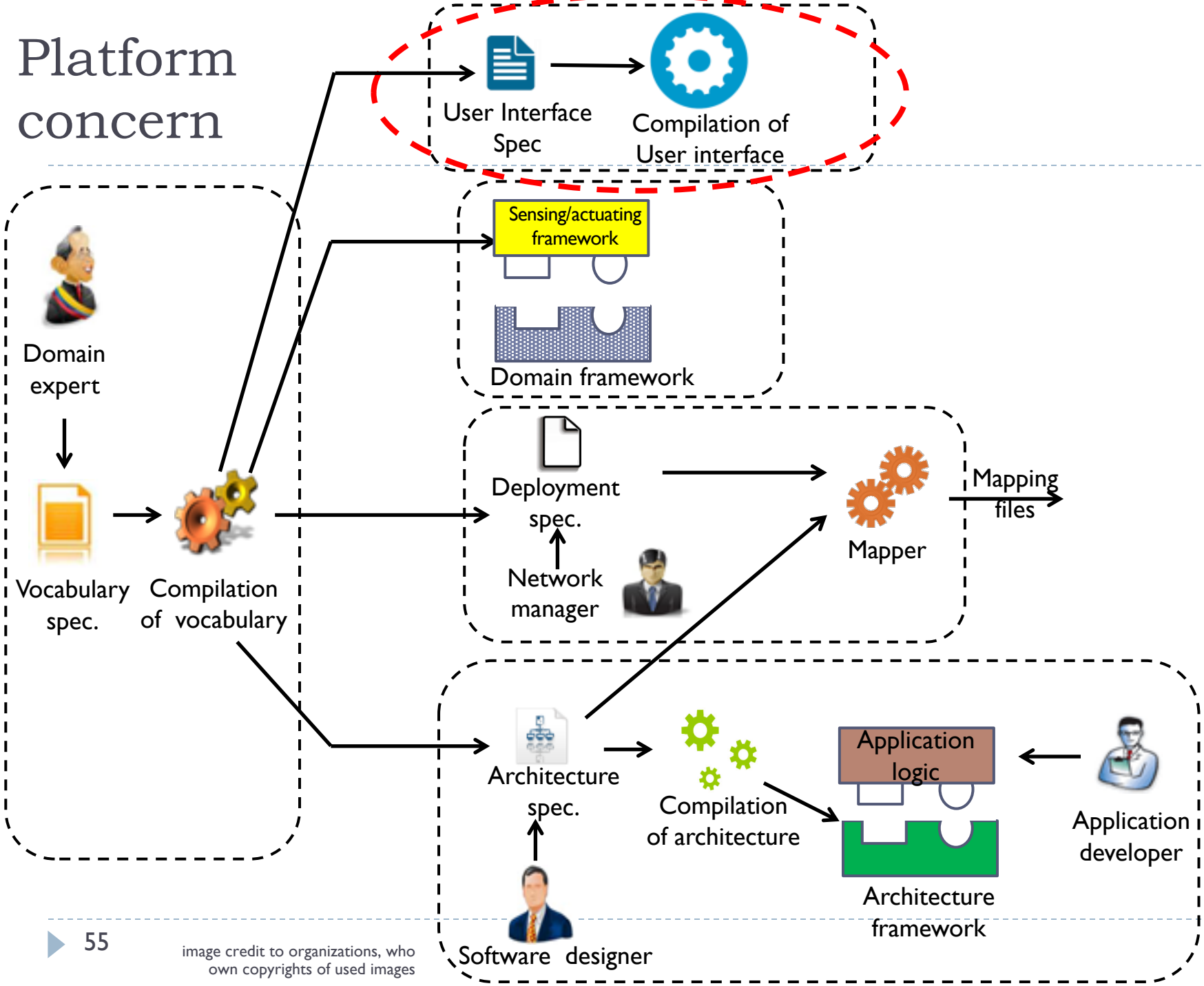
Platform concern

Concrete methods to interact with other software components & platform-specific device driver

- Sensing framework for Android
- Sensing & actuating framework for RPi in NodeJS.
- Storage Framework for MongoDB, AzureDB, MySQL

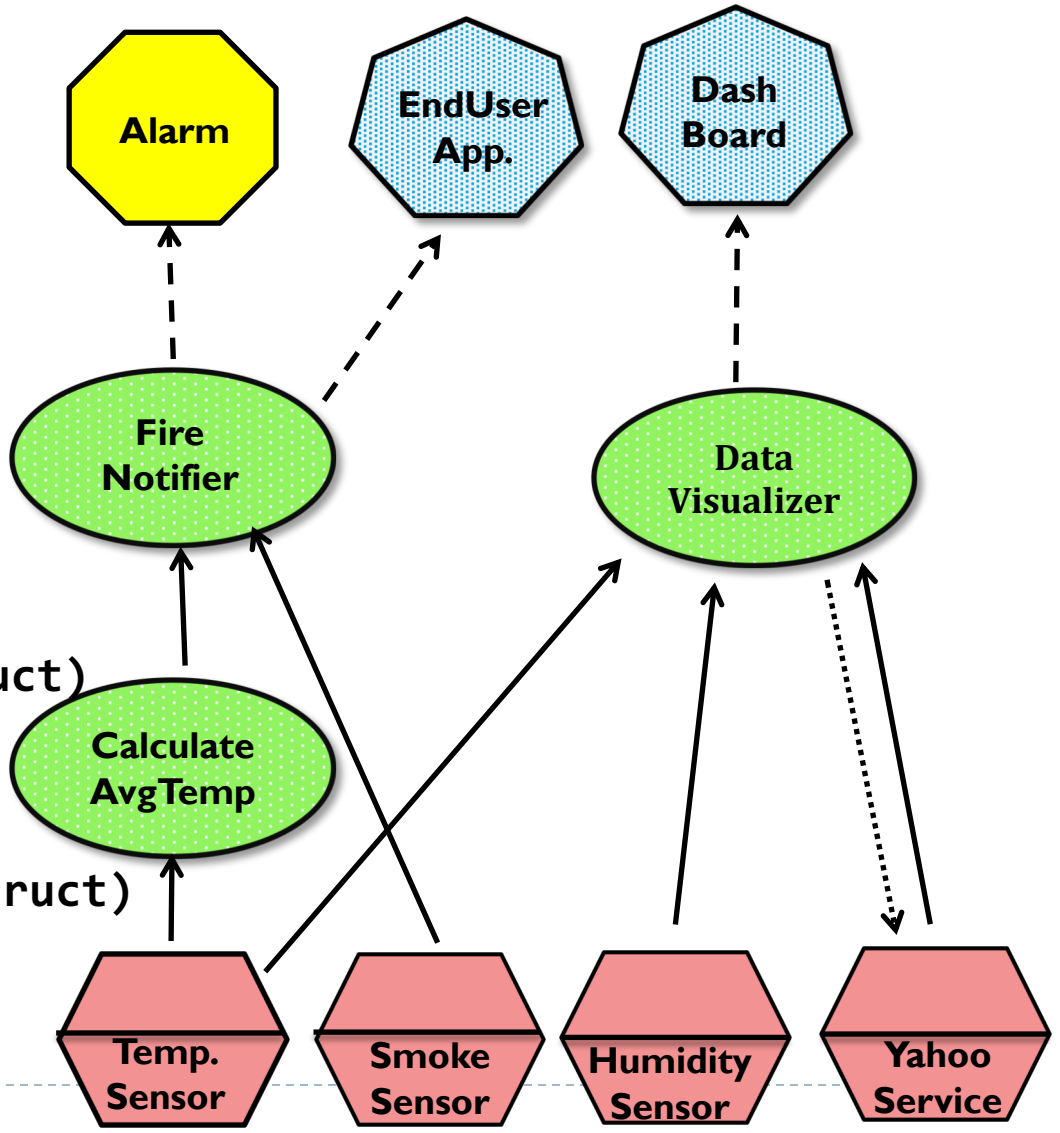


Platform concern

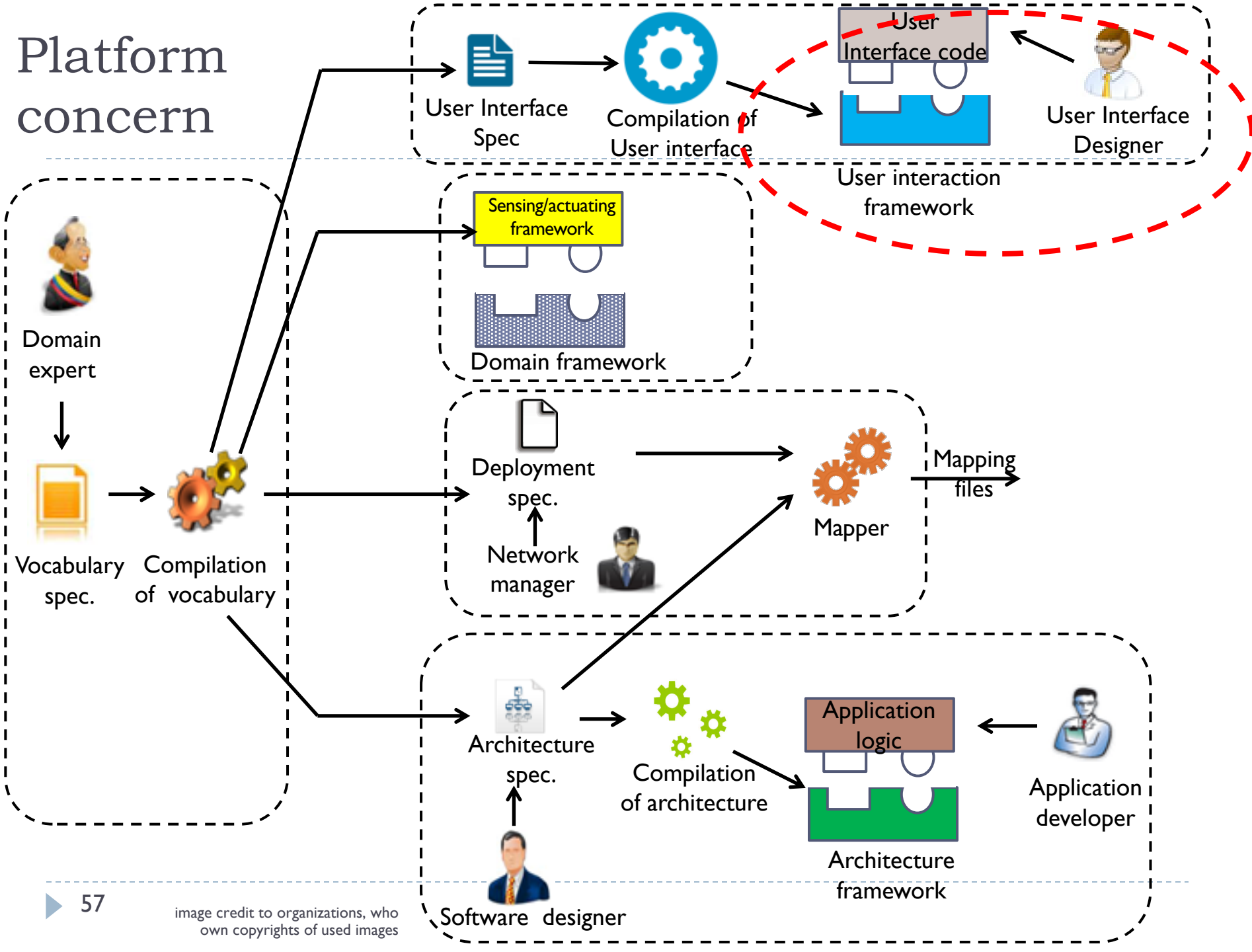


IoTSP
User interaction lang.: Code snippet

```
structs:
 VisualizeStruct
    tempValue: double;
    humidityValue: double;
    yahooTempValue: double;
 FireStruct
    fireValue: String;
    timeStamp: String;
resources:
 userInteractions:
  EndUserApp
   notify FireNotify(fn:FireStruct)
        from FireNotifier;

  DashBoard
   notify Display(sm:VisualizeStruct)
       from DataVisualizer;
```



Platform concern



IoTSP
User interaction framework: code snippet

```
EndUserApp
  notify FireNotify(fireNotify:FireStruct)from FireNotifier;
```

Compiler

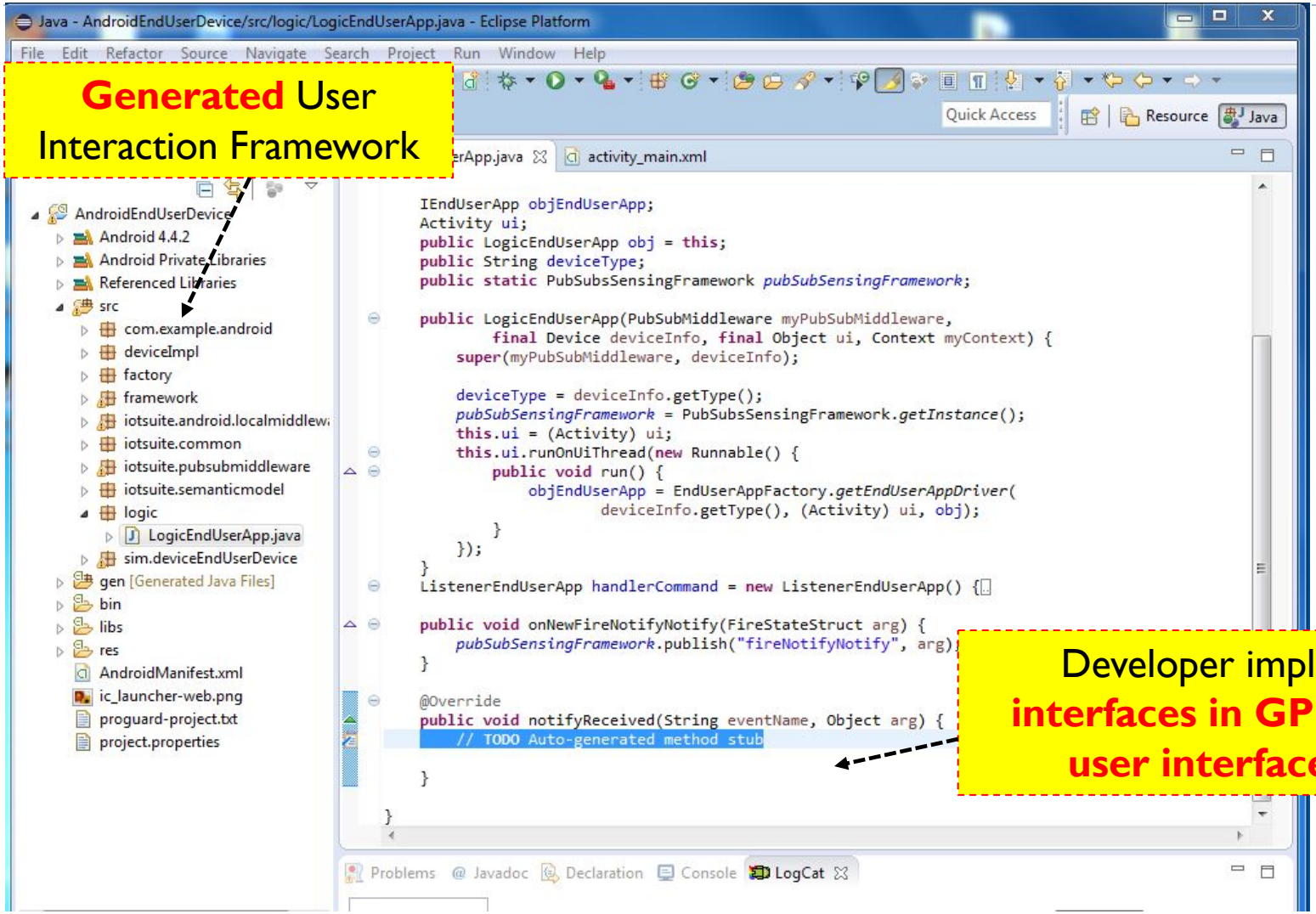
generates

User Interface Designer

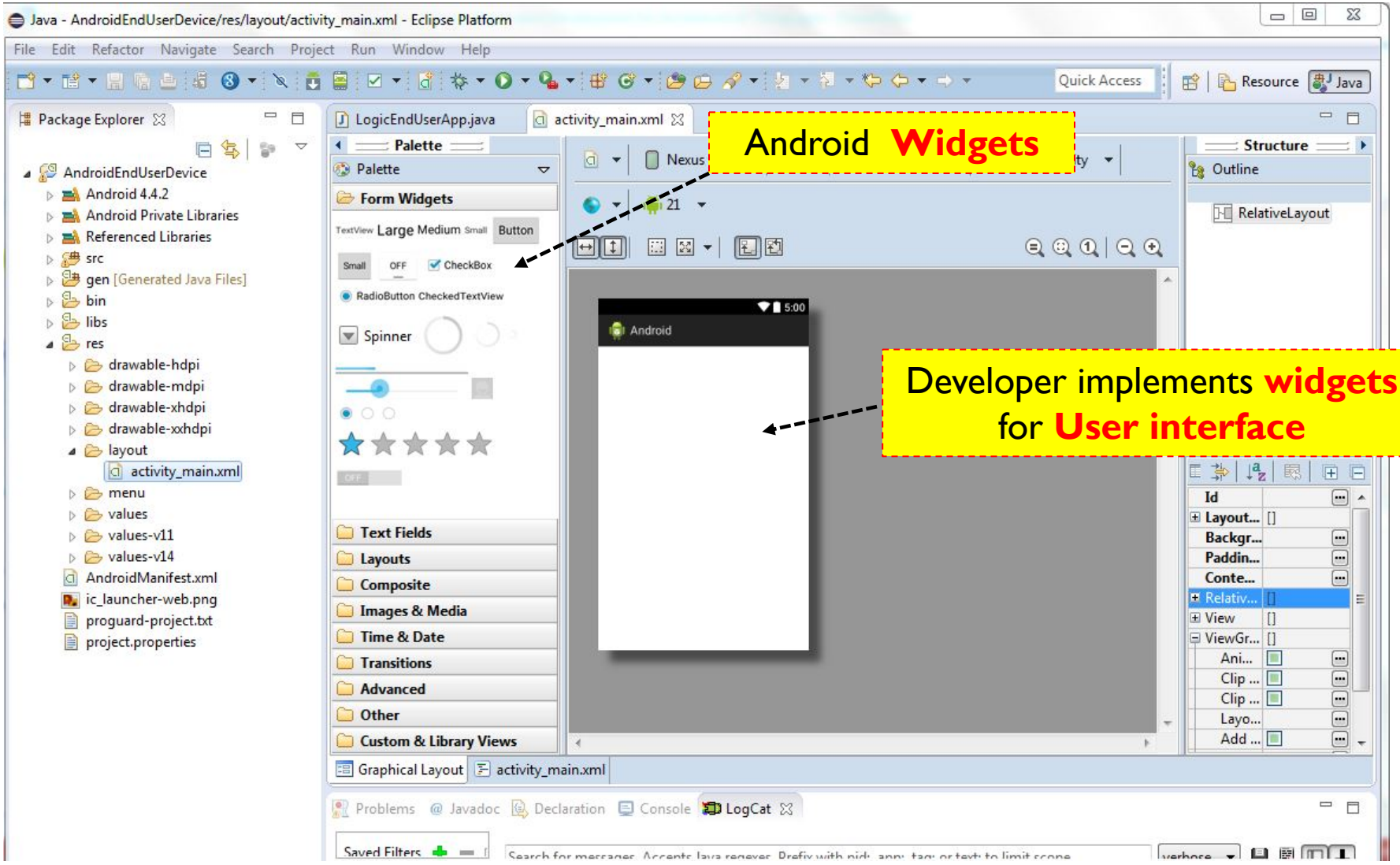
```
@Override
public void onDataReceived(String eventName, Object data) {
  if (eventName.equals("fireNotify")) {

  TextView tv = (TextView)AndroidSmartHomeApp.appActivity
              .findViewById(R.id.textView1);
  tv.setText("Fire has been occured");
 }
}
```

User interface designer implements `OnDataReceived()` method and connect them with UI widgets (e.g., Here is TextView)

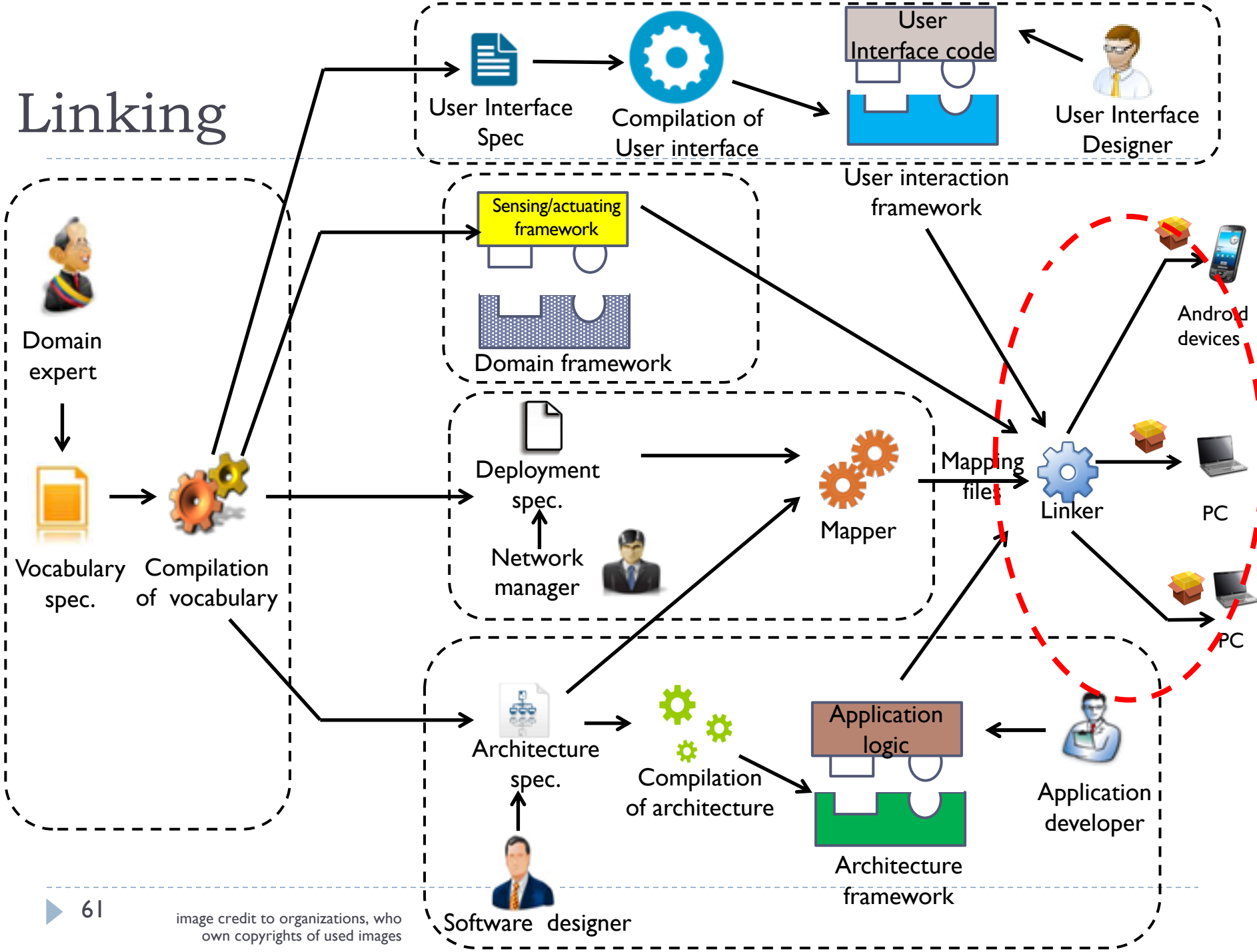
IoTSP
IoTSuite User interaction framework



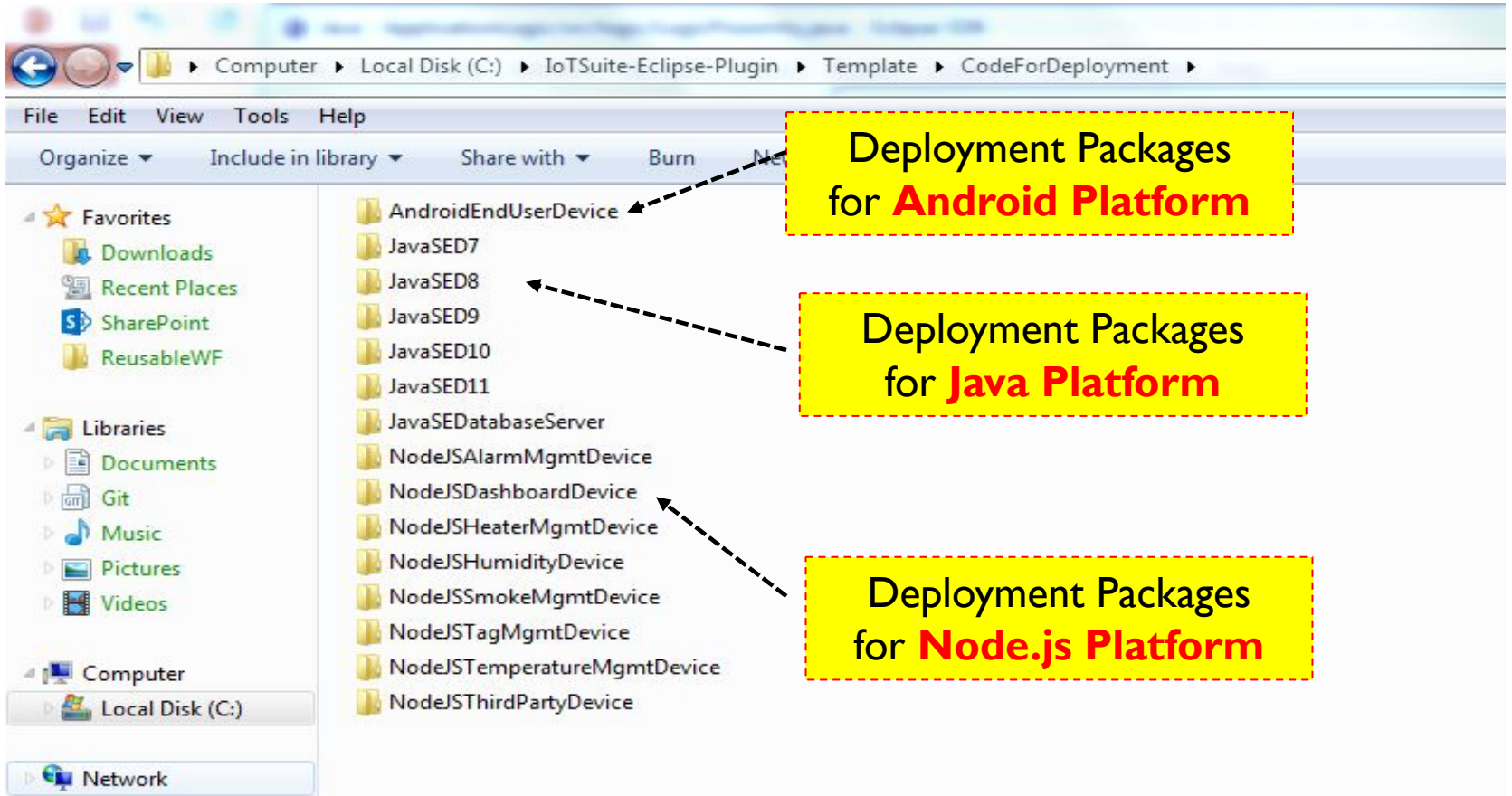
IoTSP
IoTSuite User interaction framework



Linking



IoTSP
IoTSuite Deployment packages



PackageName = [**Implementation Platform**] [**DeviceName**][“**Device**”]

Example: [**NodeJS**][**TemperatureMgmt**][“**Device**”]

State of the art: Programming IoT

Approach	Examples	Description	Benefits	Limitation
General-purpose Programming	Node.js, Python, C, C++, Android	Developers think in terms of activities of individual devices & explicitly encode interactions with others in programming language.	Development of efficient systems based on complete control over device.	More development effort, Difficult to reuse & platform-dependent design.

Reading data from Sensors

```
var sensorLib = require('node-dht-sensor');
var mqtt=require('mqtt');
var client=mqtt.connect('mqtt://test.mosquitto.org:1883');
var sensor = {
    initialize: function () {
        // here GPIO4 means pin7 and DHT22 type of sensor
        return sensorLib.initialize(22, 4);
    },
    read: function () {
        var readout = sensorLib.read();
        var value={"tempValue":readout.temperature.toFixed(2),
            , "humidityValue":readout.humidity.toFixed(2)};
        client.publish('sensorMeasurement',JSON.stringify(value));
        setTimeout(function () {
            sensor.read();
        }, 5000);
    } };
if (sensor.initialize()) {
    sensor.read(); } else { console.warn('Failed to initialize
sensor');
}
```

Connecting to MQTT Protocol

APIs to read temperature values

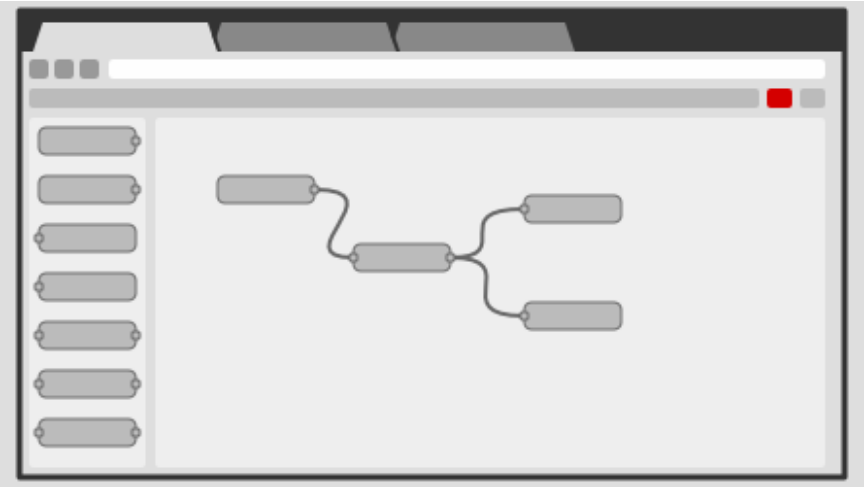
Publishing sensed value

State of the art: Programming IoT

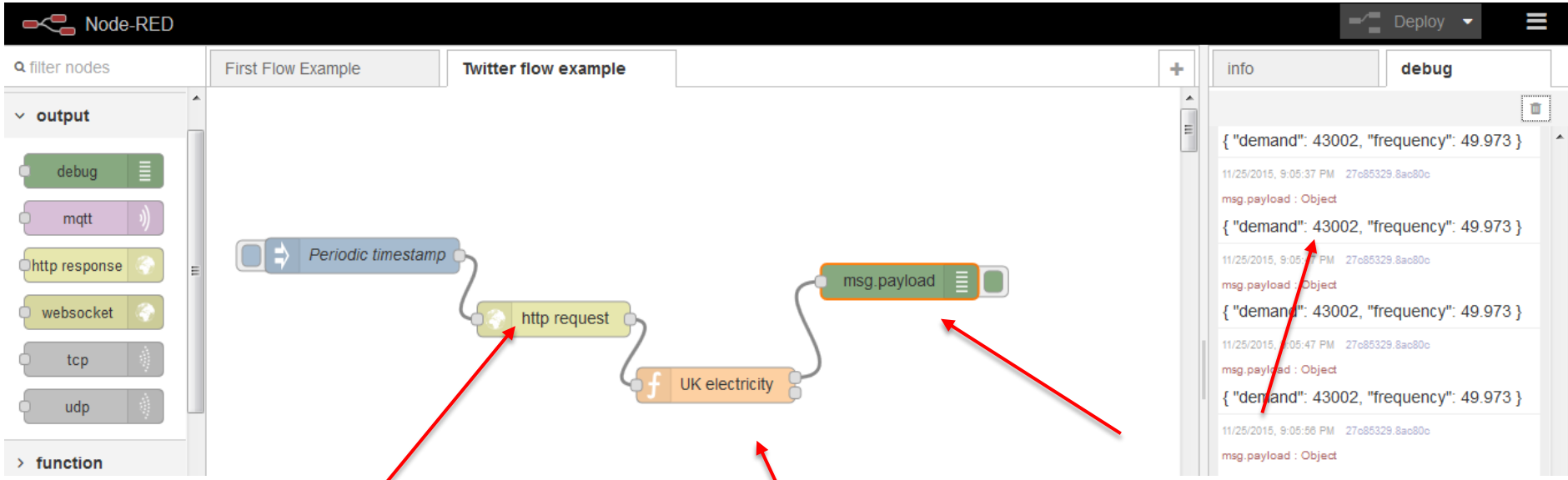
Approach	Examples	Description	Benefits	Limitation
General-purpose Programming	Node.js, Python, C, C++, Android	Developers think in terms of activities of individual devices & explicitly encode interactions with others in programming language.	Development of efficient systems based on complete control over device.	More development effort, Difficult to reuse & platform-dependent design.
Macro programming	Node-RED Regiment, MacroLab,	Abstractions to specify high-level collaborative behaviors while hiding low-level details such as message passing	Reduce development Effort compared to GPL,	Platform-specific design, Less flexible to write Customized application logic.

Rapid prototyping tool - NodeRED

- Open source tools (https://github.com/node-red)
- Flow-based programming
- Browser-based flow editor
- Invented by IBM for wiring hardware devices, APIs and online services
- Light-weight runtime such as Node.js
- Ideal to run on edge devices
- Over 2500+ ready-to-use nodes/flows https://flows.nodered.org/



Example



It displays the converted data format on Console.

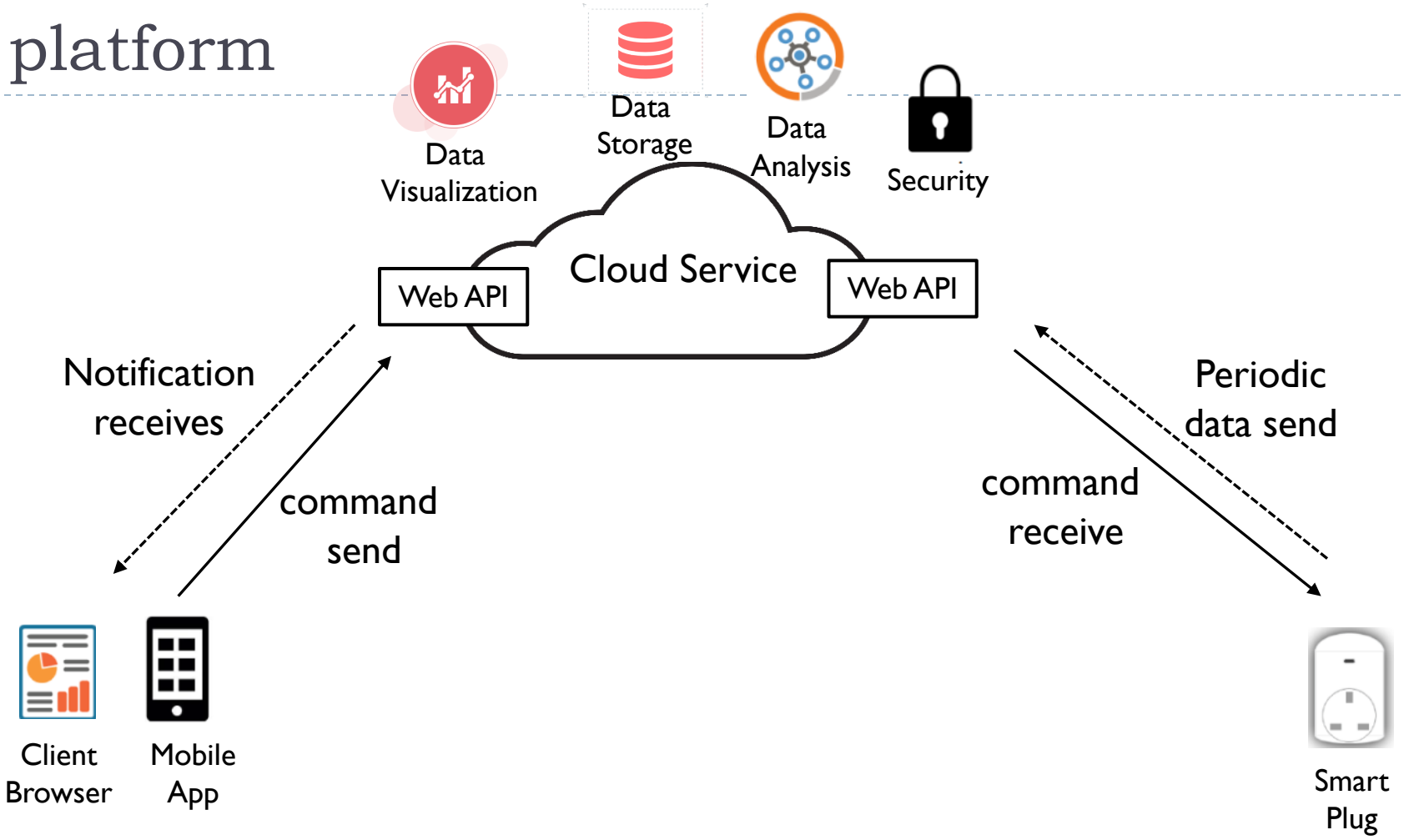
It requests data to UK national grid service periodically.

It extracts data from response and convert it to JSON format

State of the art: Programming IoT

Approach	Examples	Description	Benefits	Limitation
General-purpose Programming	Node.js, Python, C, C++, Android	Developers think in terms of activities of individual devices & explicitly encode interactions with others in programming language.	Development of efficient systems based on complete control over device.	More development effort, Difficult to reuse & platform-dependent design.
Macro programming	Node-RED Regiment, MacroLab,	Abstractions to specify high-level collaborative behaviors while hiding low-level details such as message passing	Reduce development Effort compared to GPL,	Platform-specific design, Less flexible to write Customized application logic.
Cloud Platform	IBM BlueMix, Xively, WoTKit	Devices are connected to cloud platforms through APIs or high-level (e.g., drag-and-drop) constructs, Expose common services (e.g., data visualization,) through APIs	Reduce development effort - GPL, Offer ease of deployment & evolution	Cloud-dependent design, restrict in-terms of functionality (in-network aggregation, node-to-node comm.), depend on cloud availability

Cloud platform

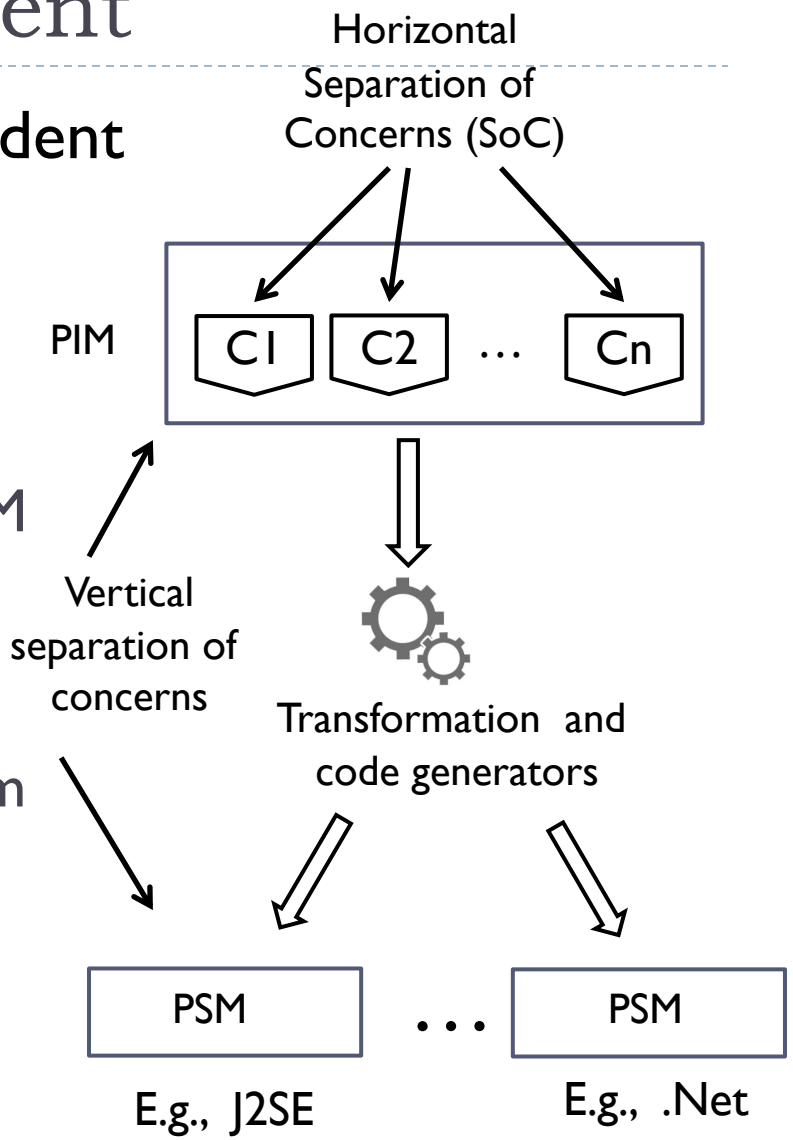


State of the art: Programming IoT

Approach	Examples	Description	Benefits	Limitation
General-purpose Programming	Node.js, Python, C, C++, Android	Developers think in terms of activities of individual devices & explicitly encode interactions with others in programming language.	Development of efficient systems based on complete control over device.	More development effort, Difficult to reuse & platform-dependent design.
Macro programming	Node-RED Regiment, MacroLab,	Abstractions to specify high-level collaborative behaviors while hiding low-level details such as message passing	Reduce development Effort compared to GPL,	Platform-specific design, Less flexible to write Customized application logic.
Cloud Platform	IBM BlueMix, Xively, WoTKit	Devices are connected to cloud platforms through APIs or high-level (e.g., drag-and-drop) constructs, Expose common services (e.g., data visualization,) through APIs	Reduce development effort - GPL, Offer ease of deployment & evolution	Cloud-dependent design, restrict in-terms of functionality (in-network aggregation, node-to-node comm.), depend on cloud availability
Model-driven Development	DiaSuite, PervML	Vertical and horizontal separation of concerns	Reusable, Extensible and Platform-independent design	Long development time to build a MDD system

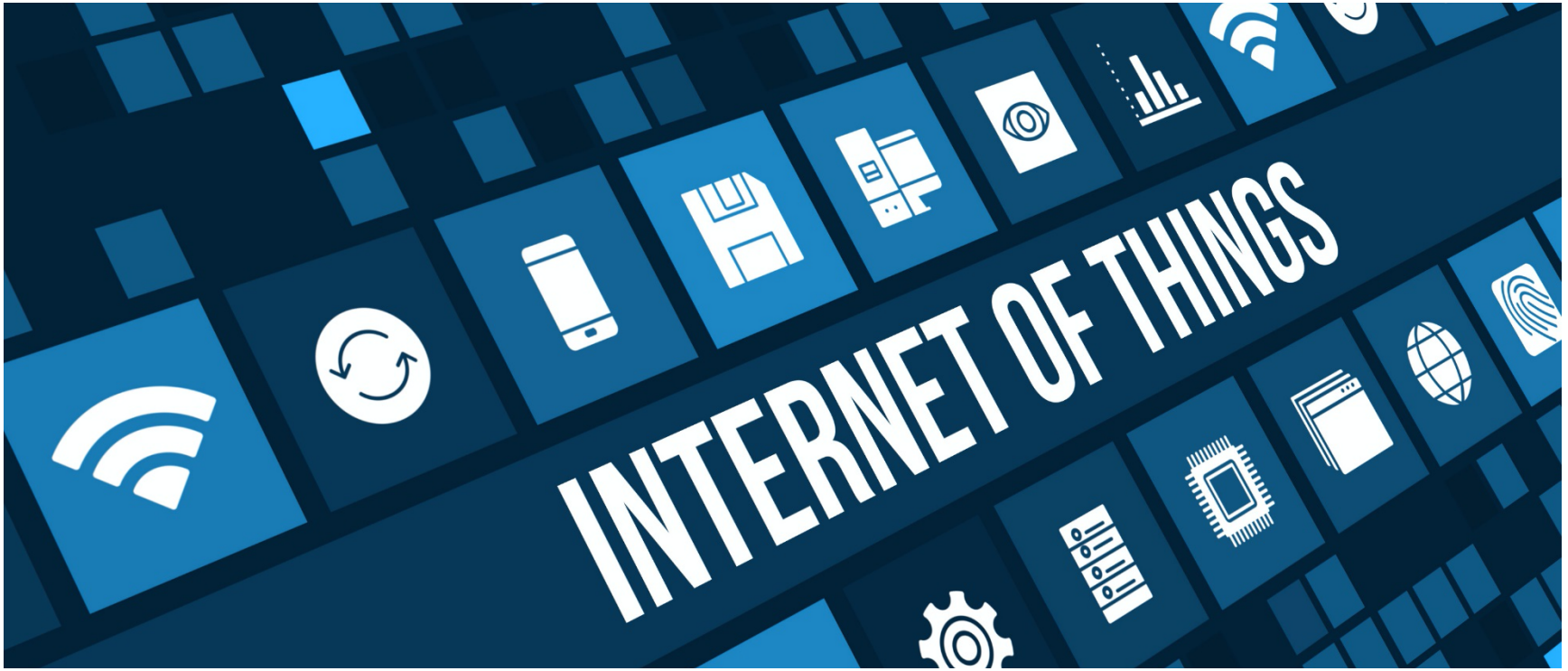
Model-driven Development

- Reusability and platform-independent
- Vertical SoC
  - Reduce application development complexity separating PIM and PSM
- Horizontal SoC
  - Separate different aspects of system



SoC – Separation of Concerns

IoTSuite: A Toolkit for Prototyping IoT Applications



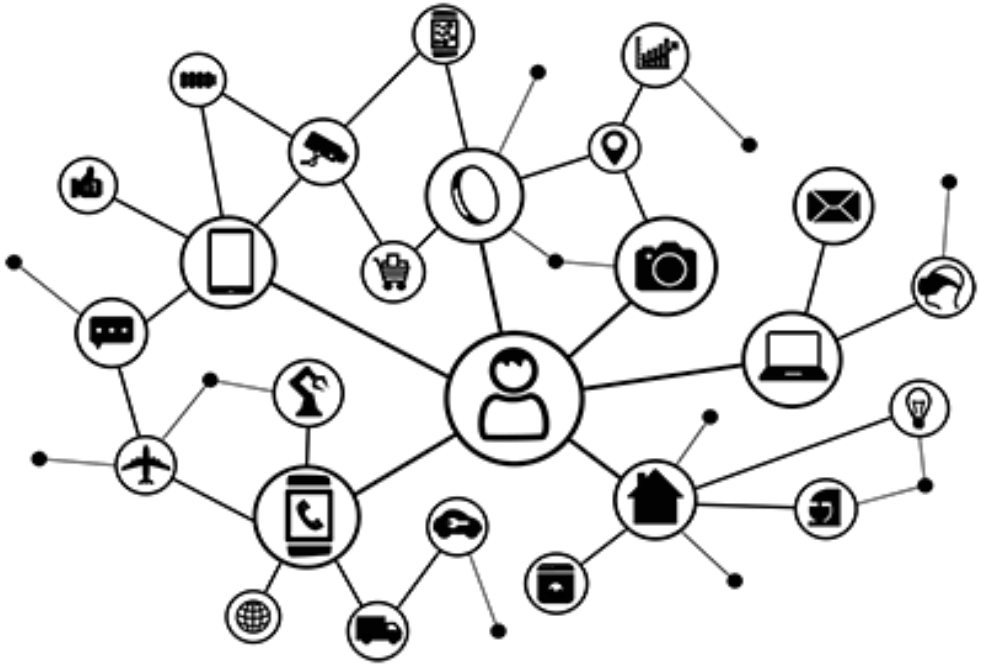
Background

- Objectives:
  - Enable IoT application development with minimal development effort

- 2010 – INRIA Paris – Rocquencourt /University of Paris VI , France
  - European Project – Large Scale Choreographies for the Future Internet (ChOReOS)
  - 2014 – PhD thesis, **Très honorable**

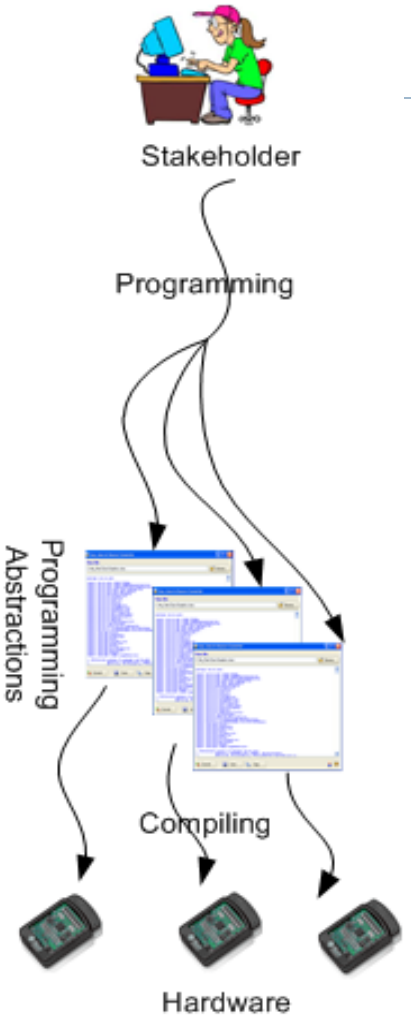
- 2014 – 2017
  - Acquire funding for new projects at ABB Corporate Research
  - Publications/Tutorials at top tire conférences (ICSE, WWW, ISWC), Mentoring B.Tech & Master thesis (at Ahmedabad University), post doc (at Insight Centre, Ireland)
  - Teaching IoT course (GCL – University of Tokyo , link)

Motivation



Heterogeneity

Different types of devices,
Platforms, Runtime systems

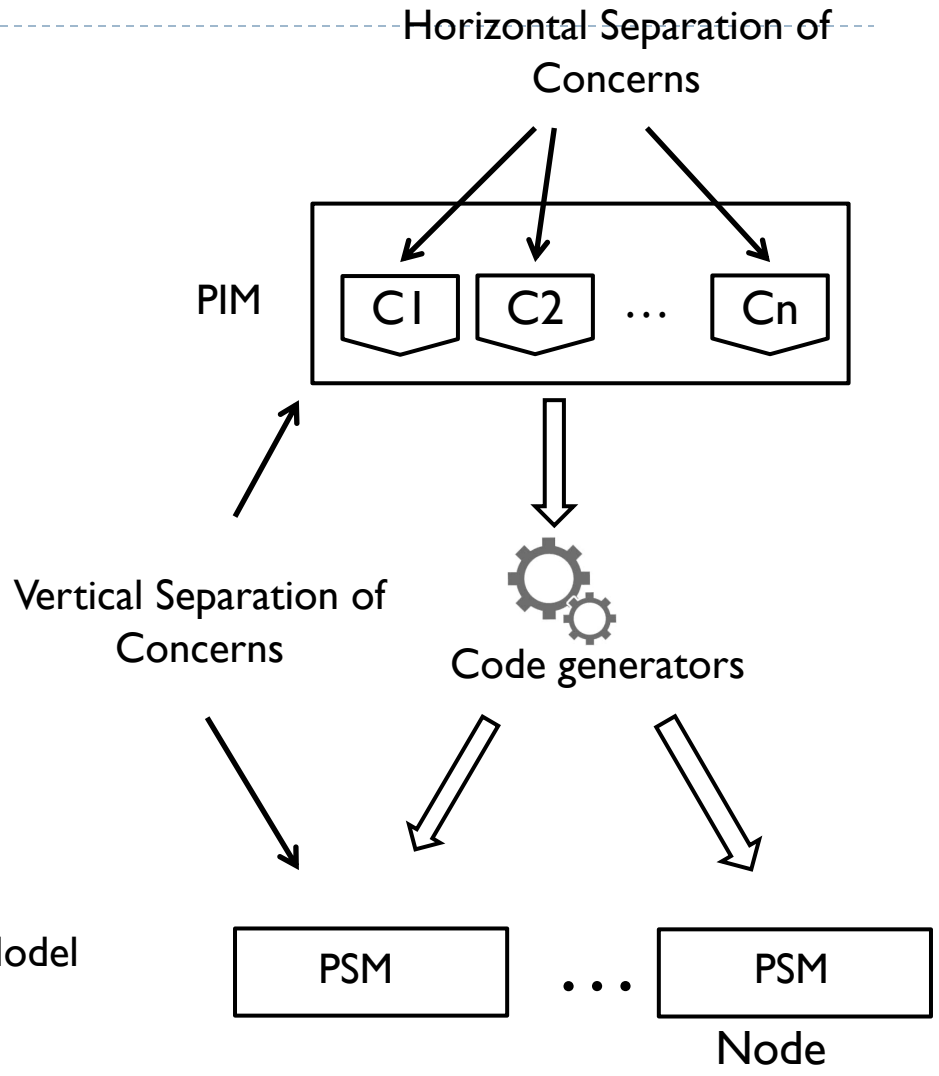


Node-centric programming

- Large number of devices

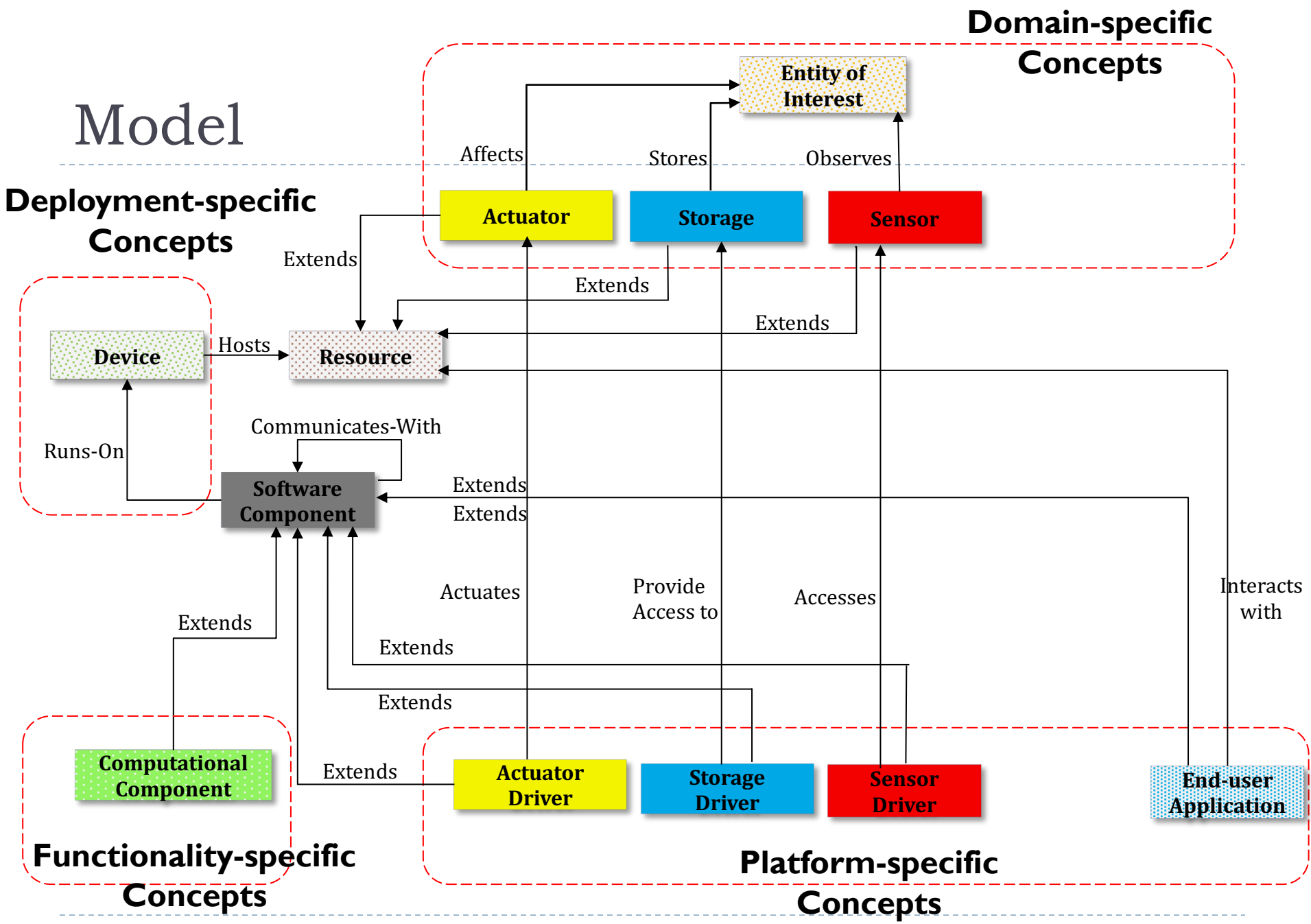
Our approach

- Separation of Concerns (reusability)
- Integration of existing DSL (reduce complexity & effort)
- Automation wherever possible (reduce effort)

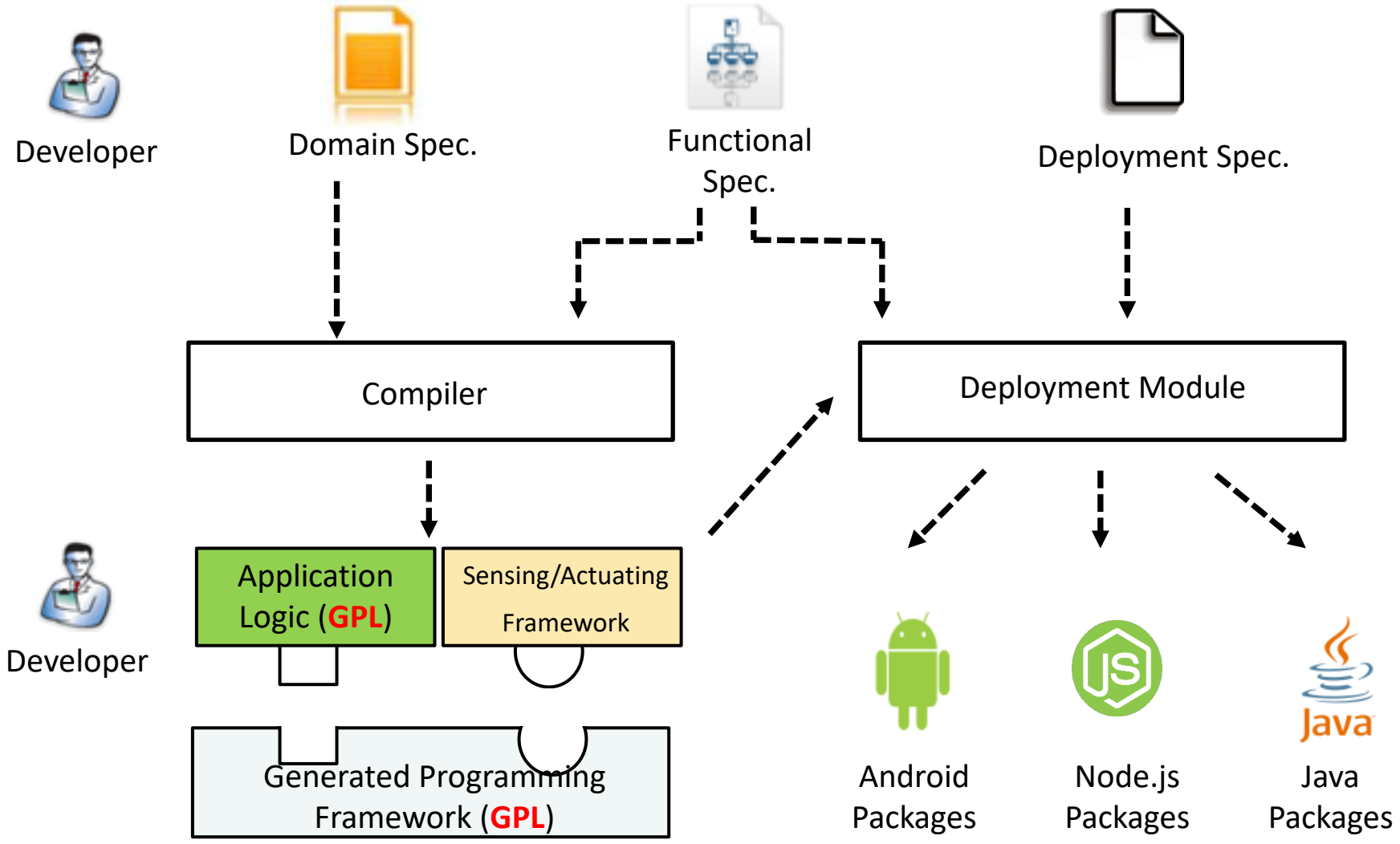


PIM – Platform Independent Model
PSM – Platform Specific Model

Model

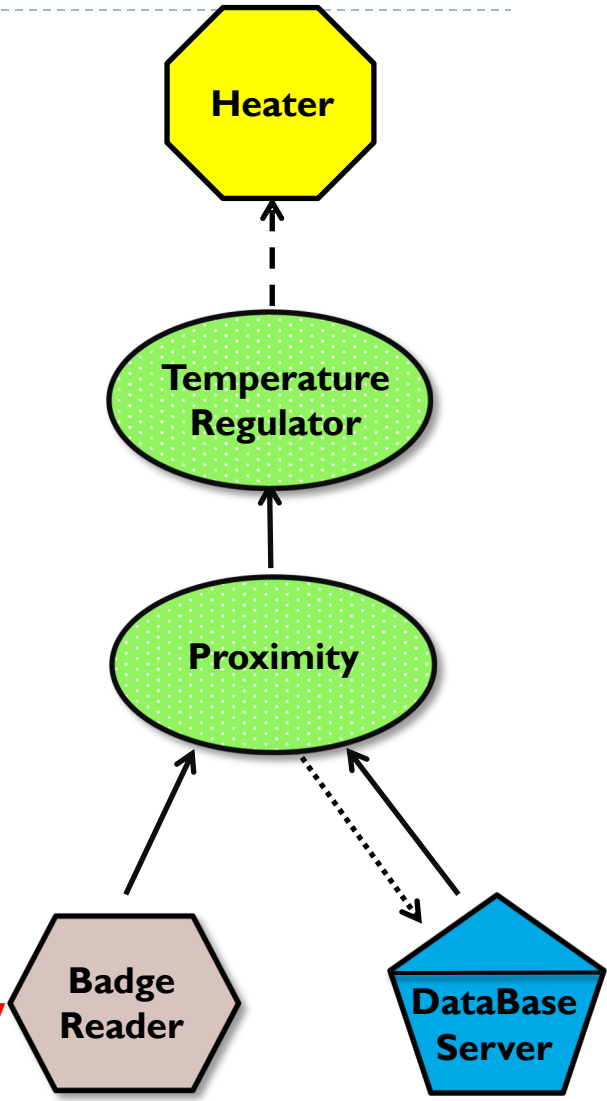

Domain-specific Concepts
Entity of Interest
Affects
Stores
Observes
Actuator
Storage
Sensor
Deployment-specific Concepts
Extends
Extends
Extends
Device
Hosts
Resource
Communicates-With
Runs-On
Software Component
Extends
Extends
Actuates
Provide Access to
Accesses
Interacts with
Extends
Extends
Extends
Computational Component
Extends
Actuator Driver
Storage Driver
Sensor Driver
End-user Application
Functionality-specific Concepts
Platform-specific Concepts

IoTSuite: Overview

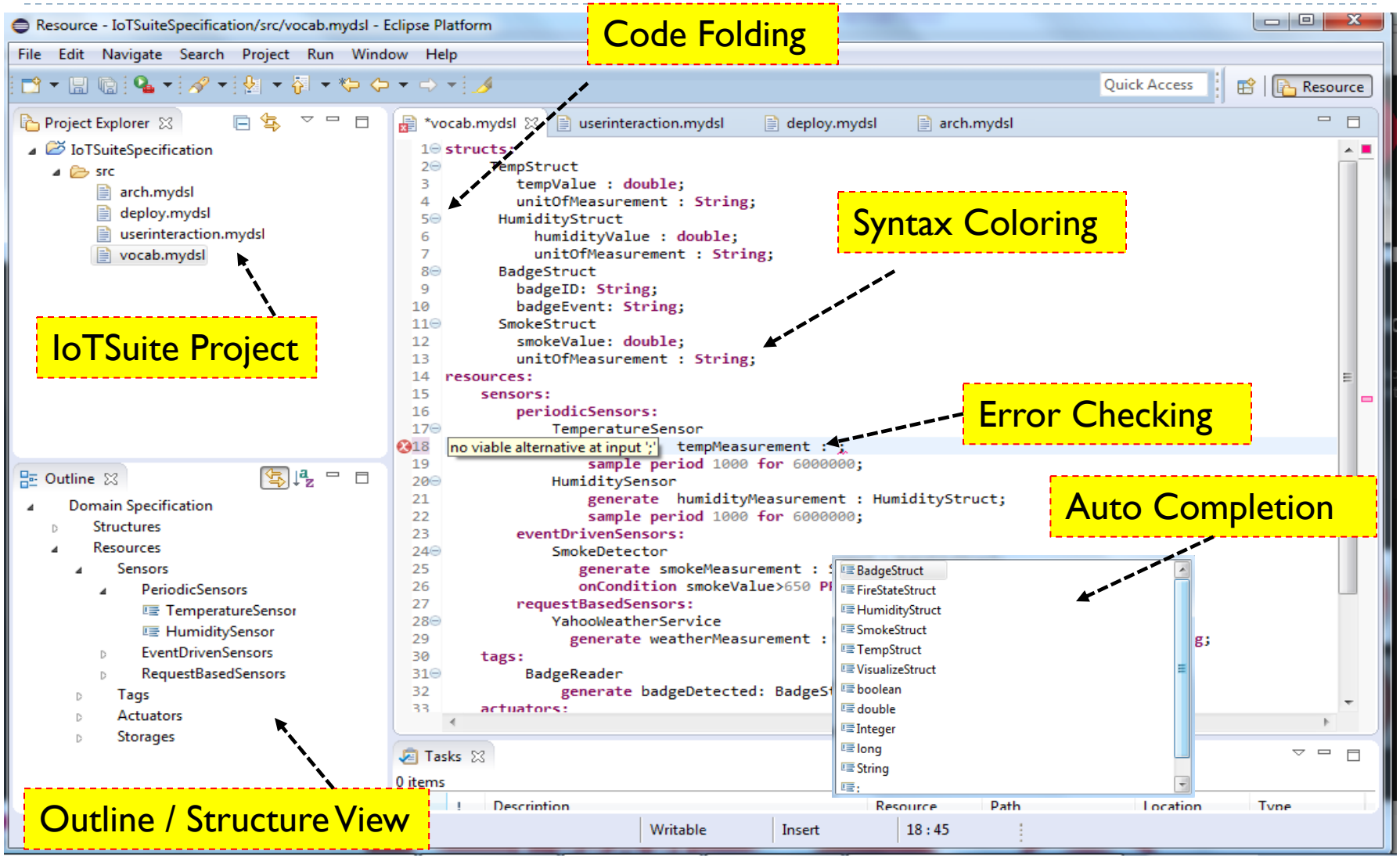


Domain language: code snippet

```
structs:
    BadgeStruct
        badgeID: String;
        badgeEvent:String;
    TempStruct
        tempValue: double;
        unitOfMeasurement: String;
resources:
    Sensors:
        BadgeReader
            generate badgeDetected: BadgeStruct;
    actuators:
        Heater
            action Off();
            action SetTemp(setTemp: TempStruct);
    storages:
        ProfileDB
        generate  profile: tempStruct accessed-by
                  badgeID: String;
```



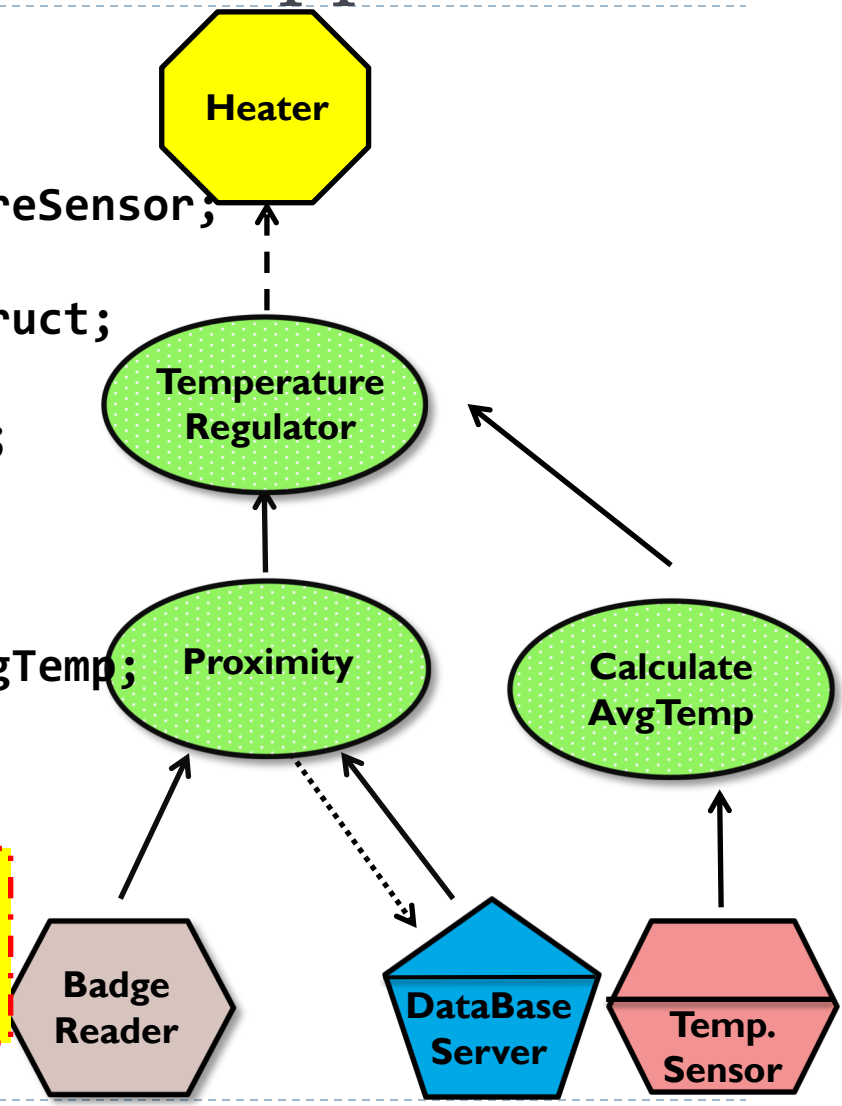
Code editor


Code Folding
Syntax Coloring
Error Checking
Auto Completion
IoTSuite Project
Outline / Structure View
Resource - IoTSuiteSpecification/src/vocab.mydsl - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Quick Access
Resource
Project Explorer
IoTSuiteSpecification
src
arch.mydsl
deploy.mydsl
userinteraction.mydsl
vocab.mydsl
*vocab.mydsl
userinteraction.mydsl
deploy.mydsl
arch.mydsl
structs:
TempStruct
tempValue : double;
unitOfMeasurement : String;
HumidityStruct
humidityValue : double;
unitOfMeasurement : String;
BadgeStruct
badgeID: String;
badgeEvent: String;
SmokeStruct
smokeValue: double;
unitOfMeasurement : String;
resources:
sensors:
periodicSensors:
TemperatureSensor
no viable alternative at input ';'
tempMeasurement :
sample period 1000 for 6000000;
HumiditySensor
generate humidityMeasurement : HumidityStruct;
sample period 1000 for 6000000;
eventDrivenSensors:
SmokeDetector
generate smokeMeasurement :
onCondition smokeValue>650
requestBasedSensors:
YahooWeatherService
generate weatherMeasurement :
tags:
BadgeReader
generate badgeDetected: BadgeSt
actuators:
Outline
Domain Specification
Structures
Resources
Sensors
PeriodicSensors
TemperatureSensor
HumiditySensor
EventDrivenSensors
RequestBasedSensors
Tags
Actuators
Storages
BadgeStruct
FireStateStruct
HumidityStruct
SmokeStruct
TempStruct
VisualizeStruct
boolean
double
Integer
long
String
Tasks
0 items
Description
Resource
Path
Location
Type
Writable
Insert
18 : 45

IoTSP
Architecture language: code snippet

```
computationalService:
 AvgTemp
  consume tempMeasurement from TemperatureSensor;
  COMPUTE (AVG_BY_SAMPLE,5);
  generate roomAvgTempMeasurement:TempStruct;
 Proximity
  consume badgeDetected from BadgeReader;
  request profile to ProfileDB;
  generate tempPref: TempStruct;
 TempController
  consume roomAvgTempMeasurement from AvgTemp;
  consume tempPref from Proximity;
  command SetTemp(setTemp) to Heater;
```

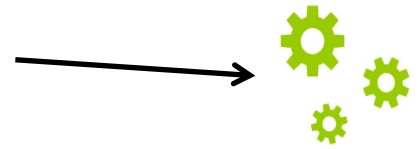
- **Programming constructs to hide heterogeneous interactions**
- **Programming constructs to perform common operations.**



Generated framework: code snippet

Structured code, Application Developer has to only implement abstract methods.

```
Proximity
  consume badgeDetected from BadgeReader;
```



Concrete method for Subscription request

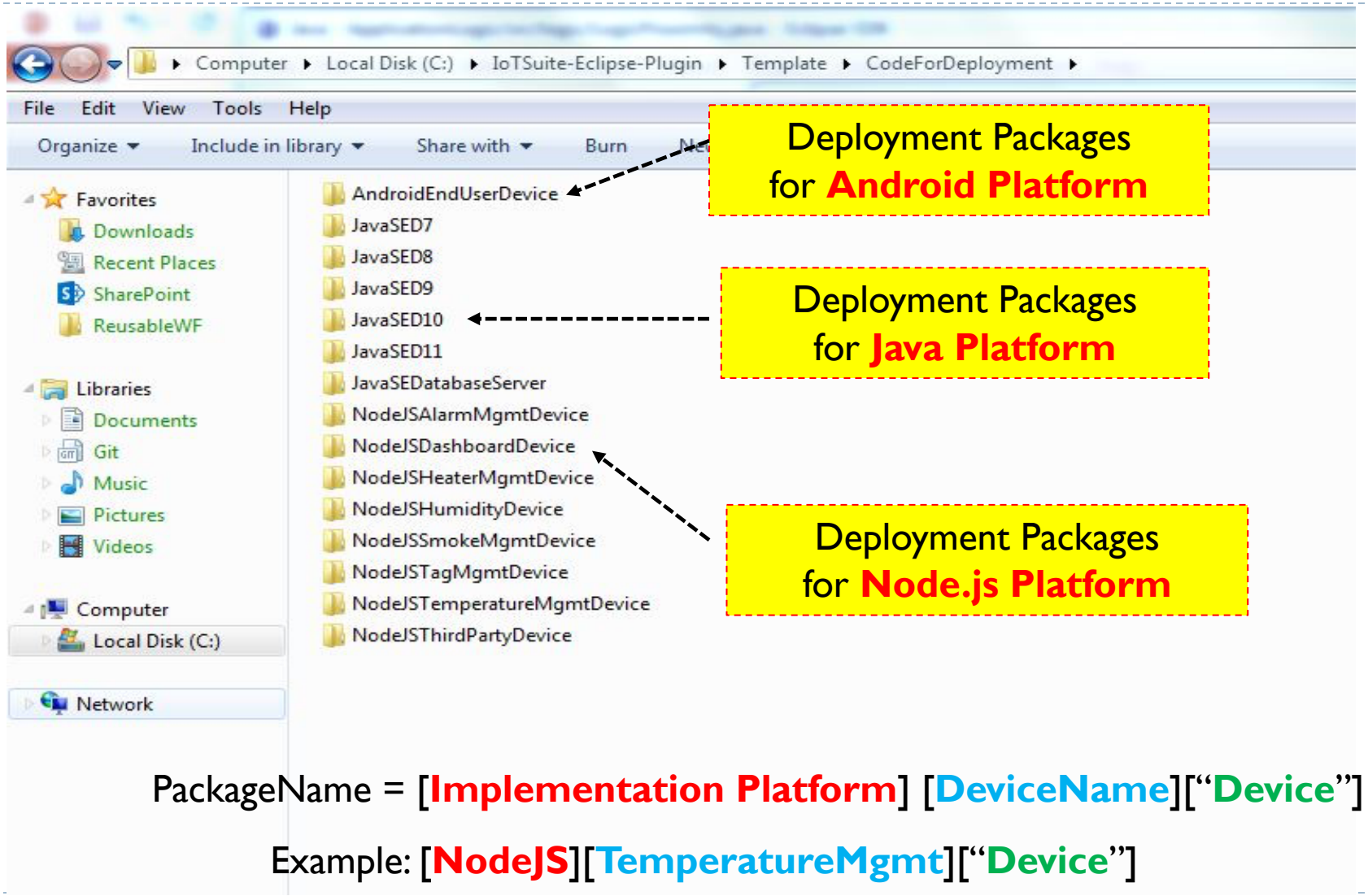
Concrete method for Receiving notification

```
public void subscribeBadgeDetected() {
        PubSubMiddleware.subscribe(this, “badgeDetected",
subscriptionCondition);
}
public void notifiedReceived (String event Name, Object
arg, Device  deviceInfo) {
      if (eventName.equals(“badgeDetected”) {
            onBadgeDetectedEvent((BadgeStruct) arg) ;
    }
}
```

Application developer

```
public abstract void onBadgeDetectedEvent(BadgeStruct arg);
```

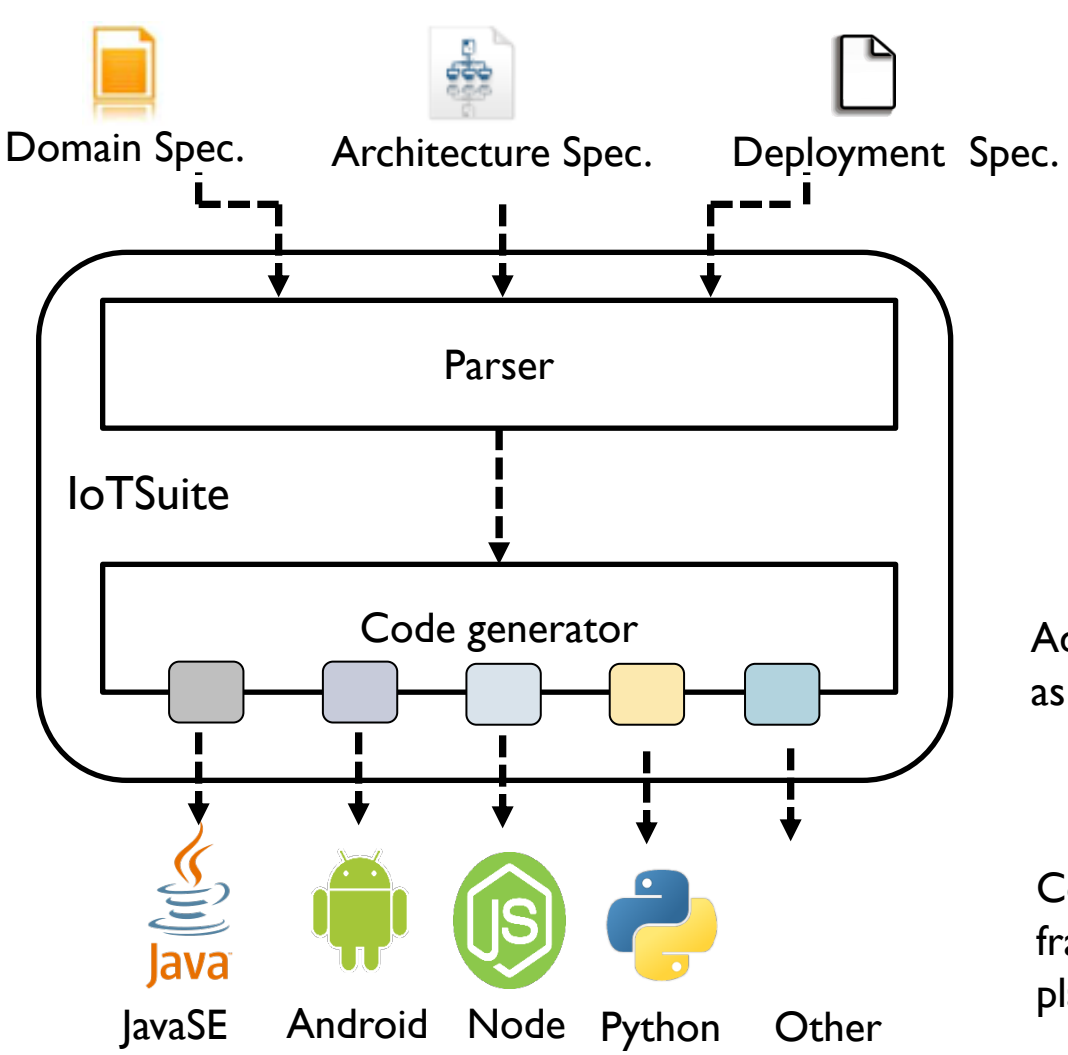
IoTSuite deployment packages



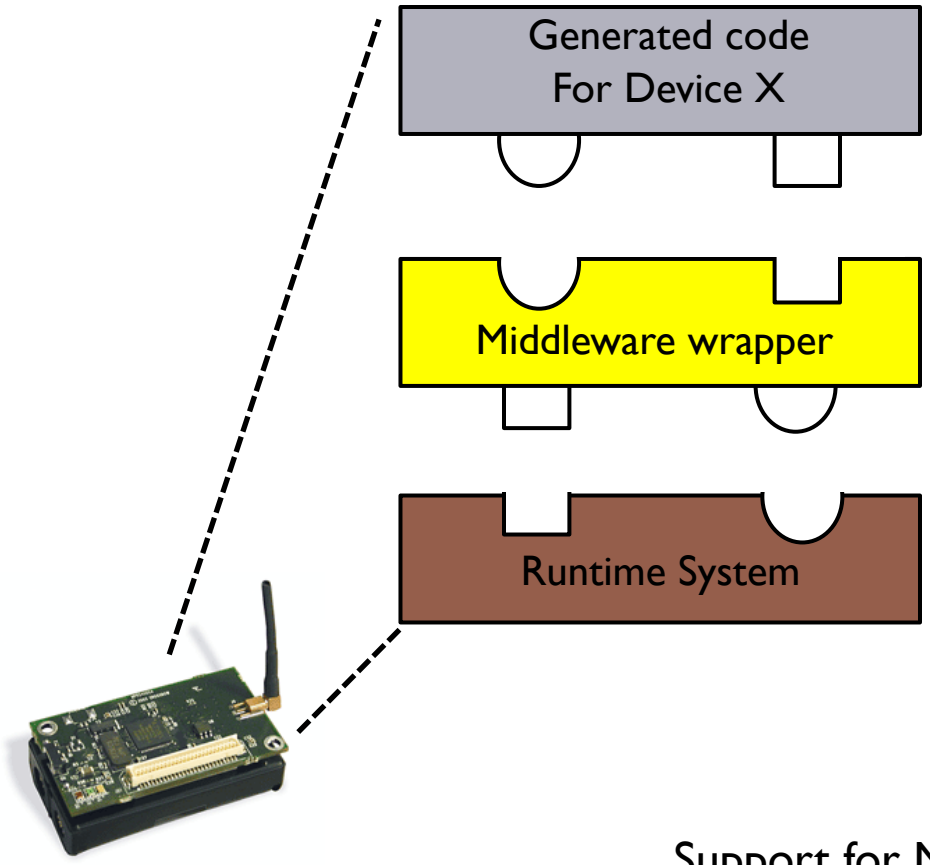
PackageName = [**Implementation Platform**] [**DeviceName**][“**Device**”]

Example: [**NodeJS**][**TemperatureMgmt**][“**Device**”]

IoTSuite: platform independent



IoTSuite: platform independent



IoTSuite generates code for a device

It plugs "generated code for a device" & runtime system. It implements interface specified in a support library, specific to a runtime system.

It runs on each individual device & provide support for executing distributed tasks.

Support for MQTT & iBICOOP

Open source

IoTSuite code is available at
https://github.com/pankeshlinux/IoTSuite

The current version of IoTSuite supports several IoT technologies such as Android, Raspberry PI, Arduino, and JavaSE-enabled devices, Messaging protocols such as MQTT, CoAP, websocket, Server technologies such as node.js, Relational database such as MySQL, and Microsoft Azure Cloud services.

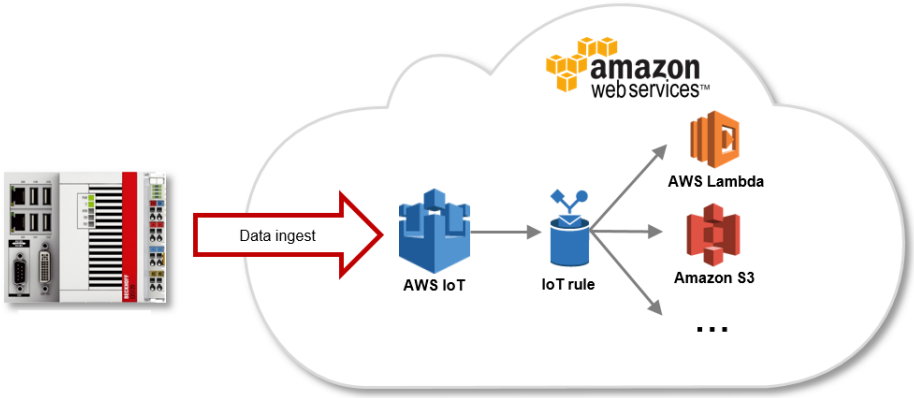
Confirm
Smart Manufacturing

Building a Data Anaytics Platform Using Open Source Tools

Ali Intizar, Pankesh Patel, John Breslin (NUI Galway)

State of the art: IoT in Cloud

- Rapid innovation and prototyping
  - Reduce time to market
  - New features and offering
  - Use of cutting edge technologies
  - Lower upfront IT costs



Microsoft Azure

https://bit.ly/2KLKbs9

aws

https://amzn.to/2DMqgLm

ORACLE

https://bit.ly/2xTcwZQ

ptc

https://bit.ly/2QoFl7v

SAP

https://bit.ly/2NfjLAh

PREDIX

https://www.predix.io

SIEMENS

https://bit.ly/2DMc0CB

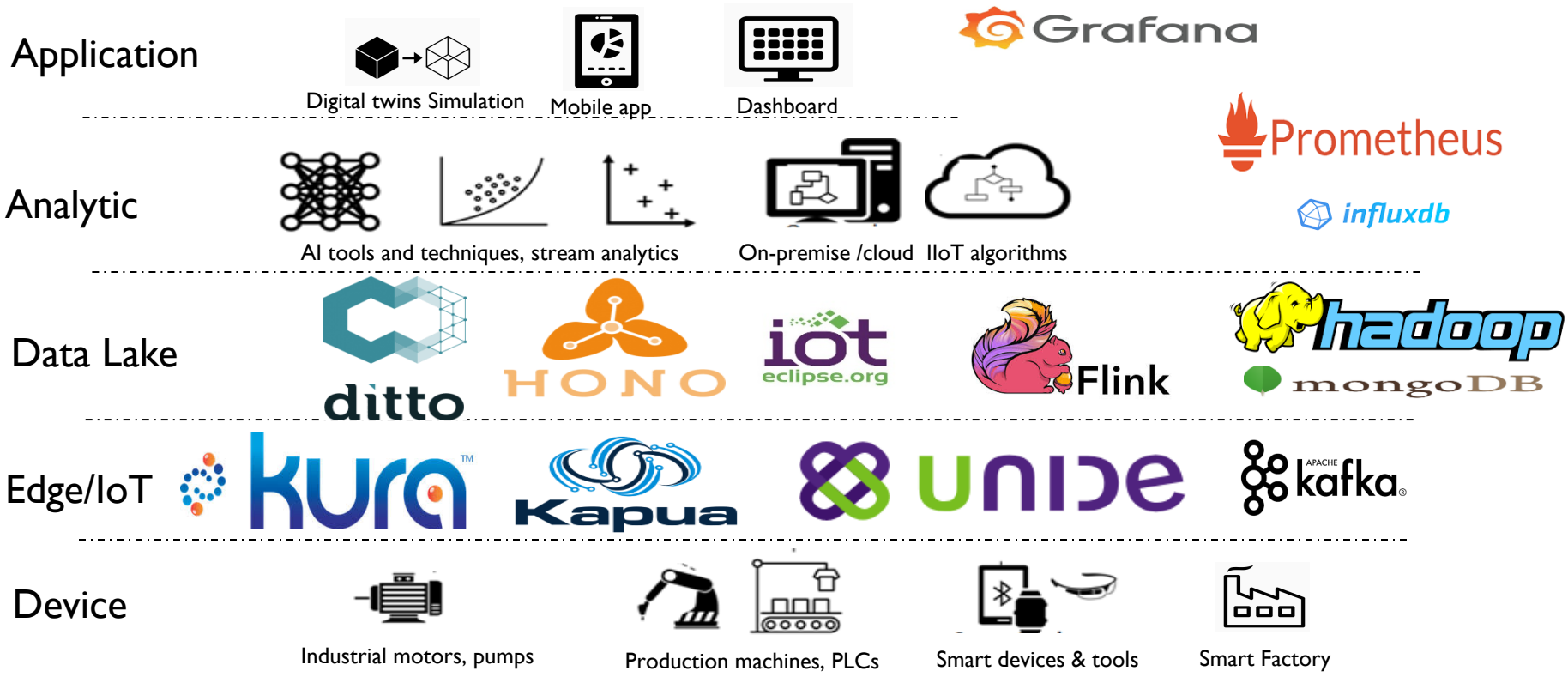
Motivation – cloud provider limitations

- Not open source!
- Freedom of choice
  - On-premise or in the cloud
- Platform-specific development design
  - Migrating from one (e.g., Microsoft Azure IoT hub) to the other cloud (e.g., AWS IoT) could be complex
  - Different application APIs to be used for different clouds
- Vendor lock-in
  - If a cloud provider increase its pricing model, it could kill the overall business revenue.
- Innovation

Open source tools

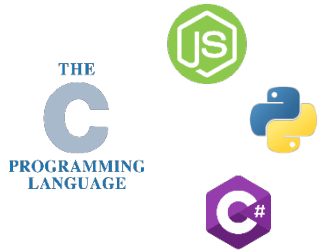
- Eclipse Hono: A set of docker based micro services
- Eclipse Unide: Production Performance Measurement Protocol
- Eclipse Kura: Analytics on Edge devices
- Eclipse Kapua: IoT Platform for sensors data managed
- InfluxDB/Prometheus: A database to store Industrial IoT data
- Grafana: Visual interface and Dashboard for Smart Factory

Early result: technology stack



Icon source: https://thenounproject.com

Summary



Programming Languages

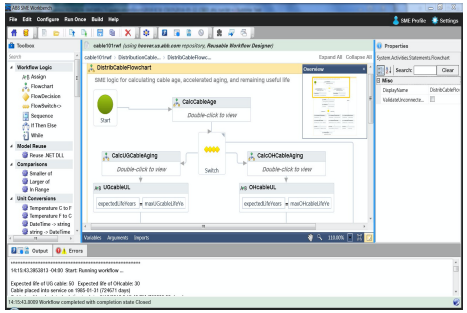
- Full control on AL (app logic)
- More development effort

Rapid prototyping tools

- Reduce development effort
- Platform-specific design (Language, Runtime)

Cloud

- Reduce development effort, ease of deployment & evolution (due to centralized system)
- Cloud-dependent design

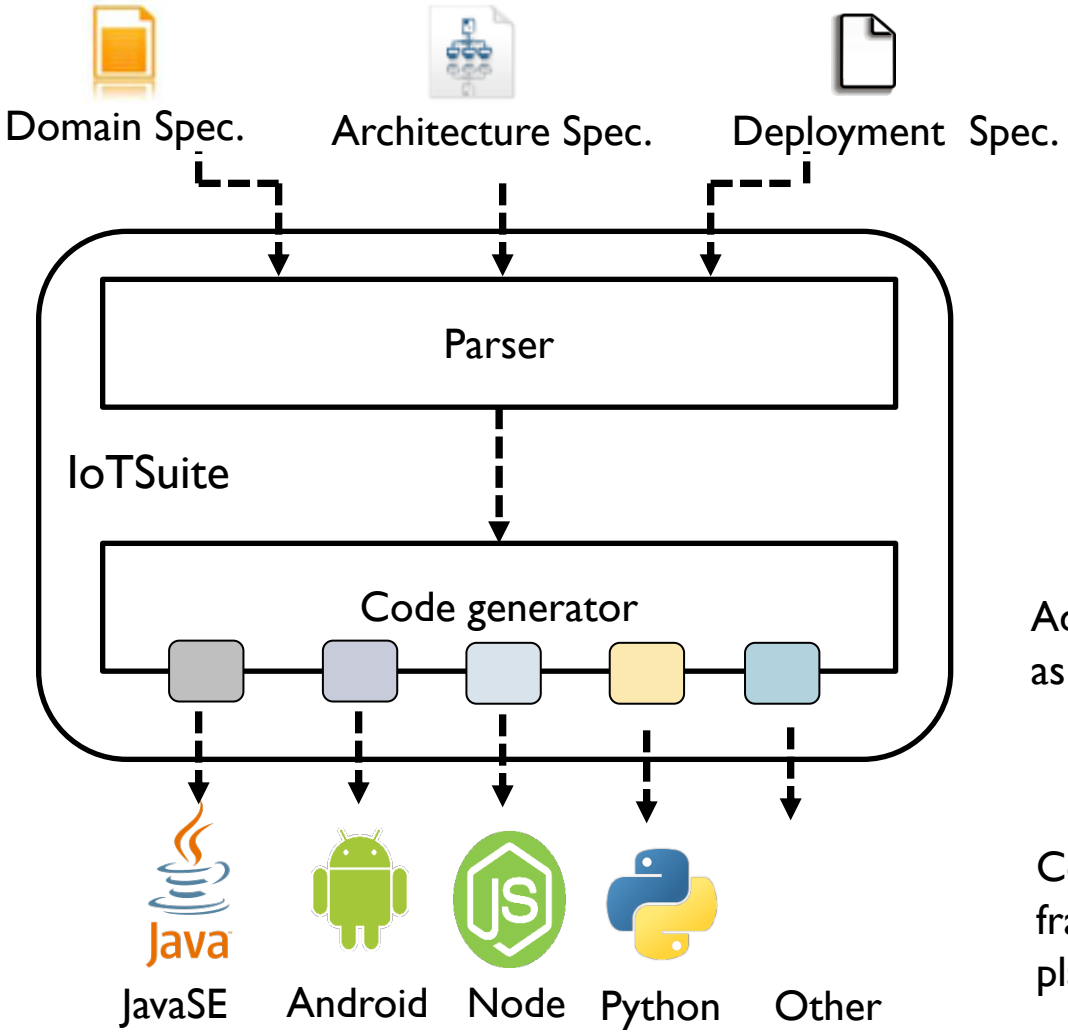


MDD

- PIM, PSM
- Longer development time due to complex and very domain specific

IoTSuite: platform independent

System specification (Platform independent)



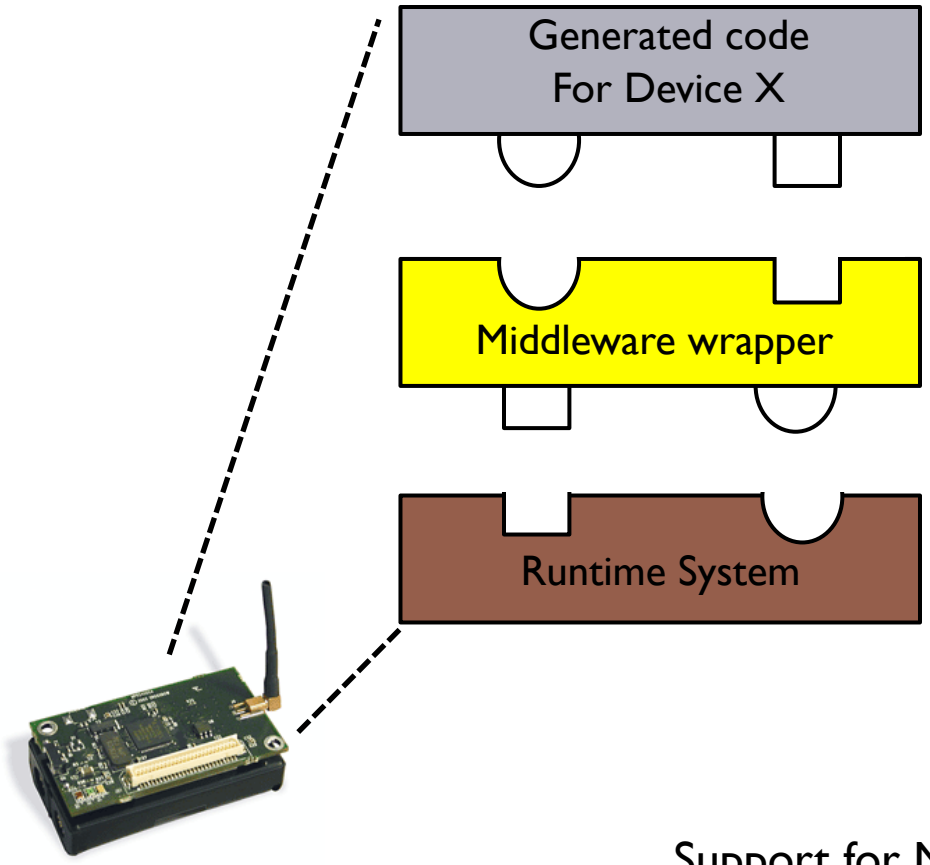
ANTLR, a parser generator from a grammar

StringTemplate, a template engine for generating source code

Adding a new platform as a plugin

Code generation of a framework in a target platform

IoTSuite: platform independent



IoTSuite generates code for a device

It plugs "generated code for a device" & runtime system. It implements interface specified in a support library, specific to a runtime system.

It runs on each individual device & provide support for executing distributed tasks.

Support for MQTT & iBICOOP

Open source

IoTSuite code is available at
https://github.com/pankeshlinux/IoTSuite

The current version of IoTSuite supports several IoT technologies such as Android, Raspberry PI, Arduino, and JavaSE-enabled devices, Messaging protocols such as MQTT, CoAP, websocket, Server technologies such as node.js, Relational database such as MySQL, and Microsoft Azure Cloud services.

Thank you for your attention

Questions