Abstractions for Event Driven Programs

Christophe De Troyer
Jens Nicolay, Christophe Scholliers, Wolfgang De Meuter

Running Example

Running Example

Running Example

Running Example

Programming Walking Lights: requirements

- Communicate through different channels
- Detect presence of lights
- Control lights that are disconnected
- Use different types of lights

Programming Walking Lights: hardware

Low resource

Bluetooth & WiFi

Low level software

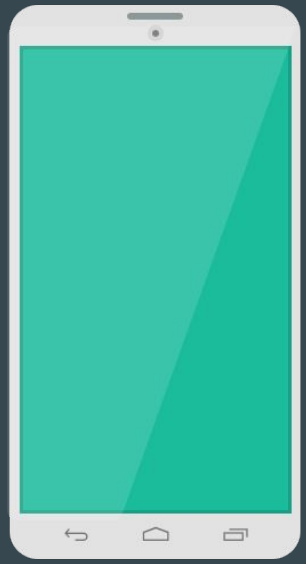
Firmware

High resource

Multiple interfaces

Full OS

VM Possible



Sending bytes back and forth

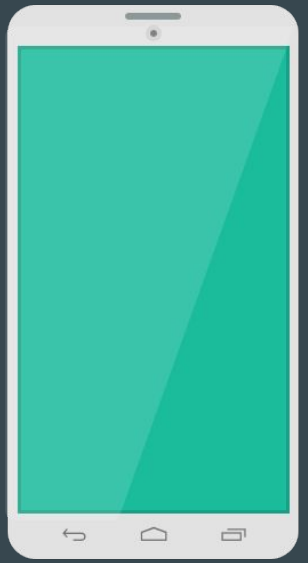
...

Discovery

Pair to WiFi

WiFi broadcast

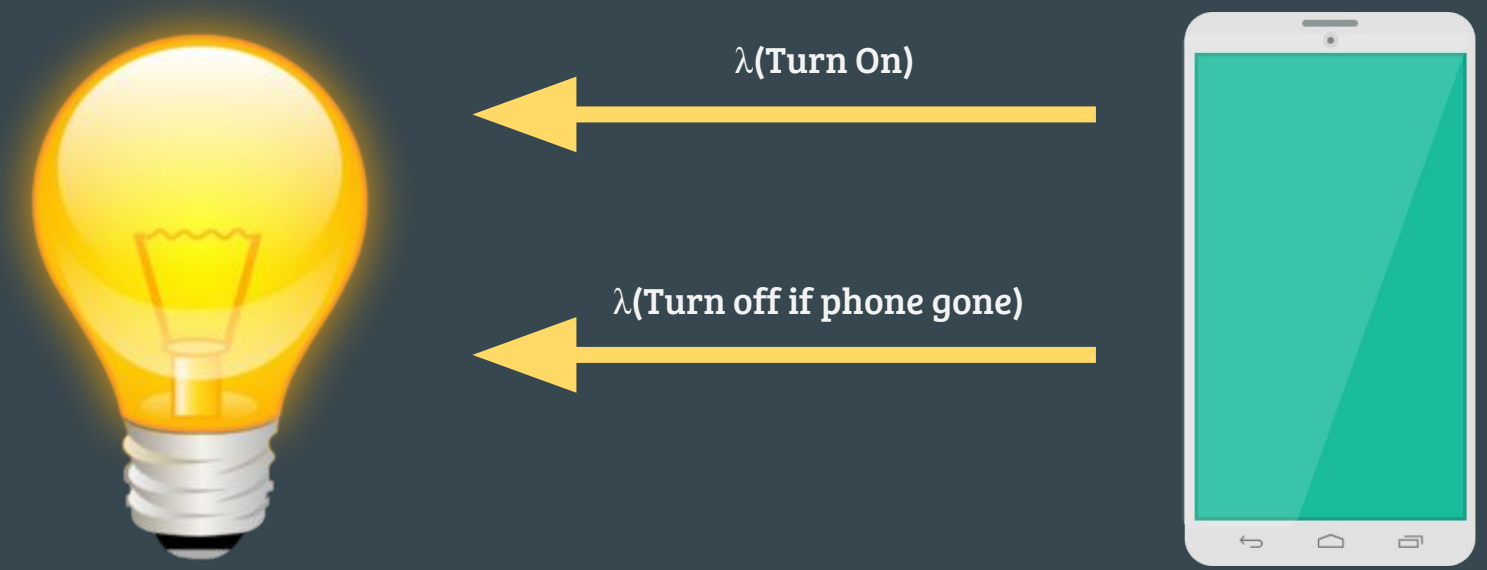
Bluetooth broadcast



DISCOVER LAMP#123

NEAR LAMP#123

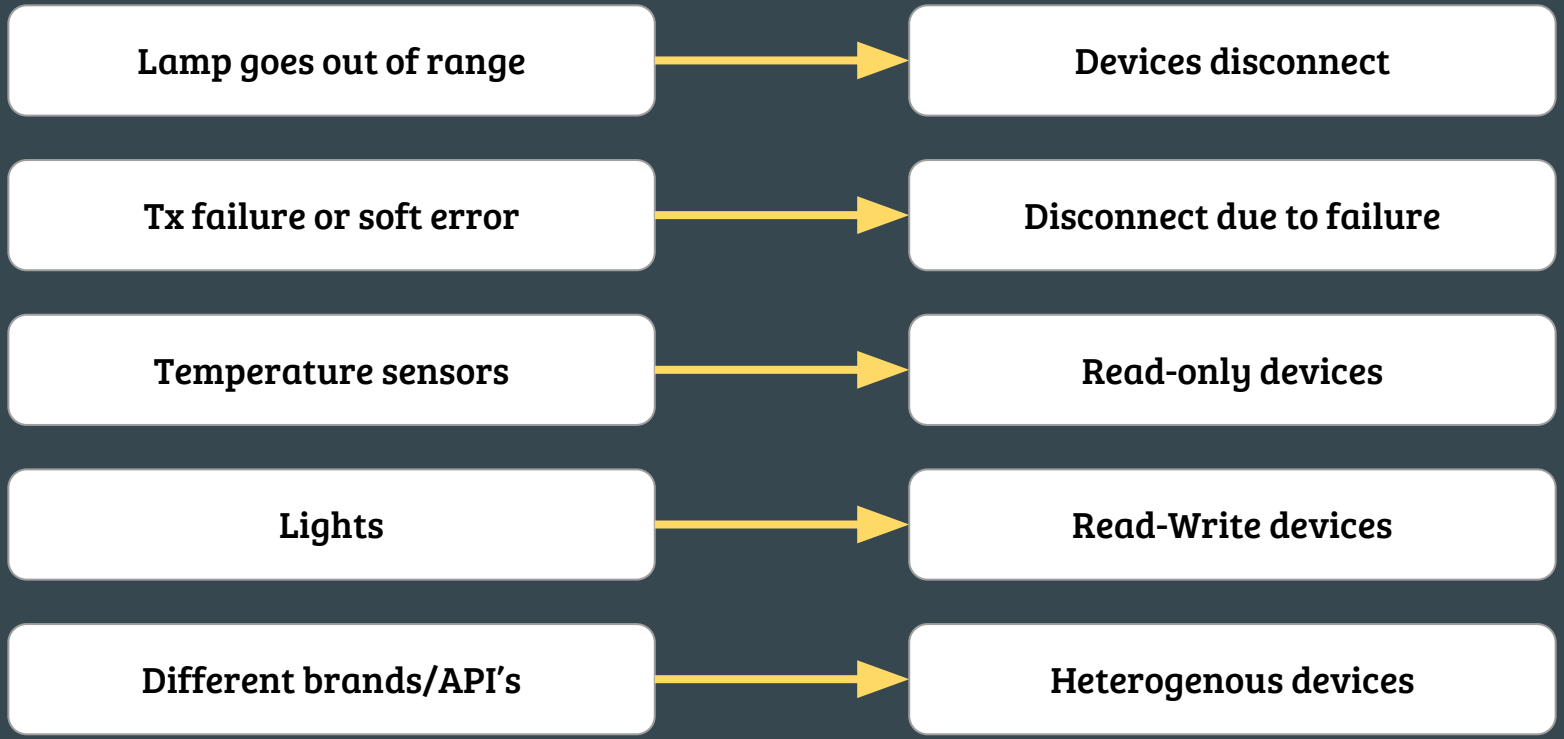
Communication



Dealing with the network

...

Network Properties



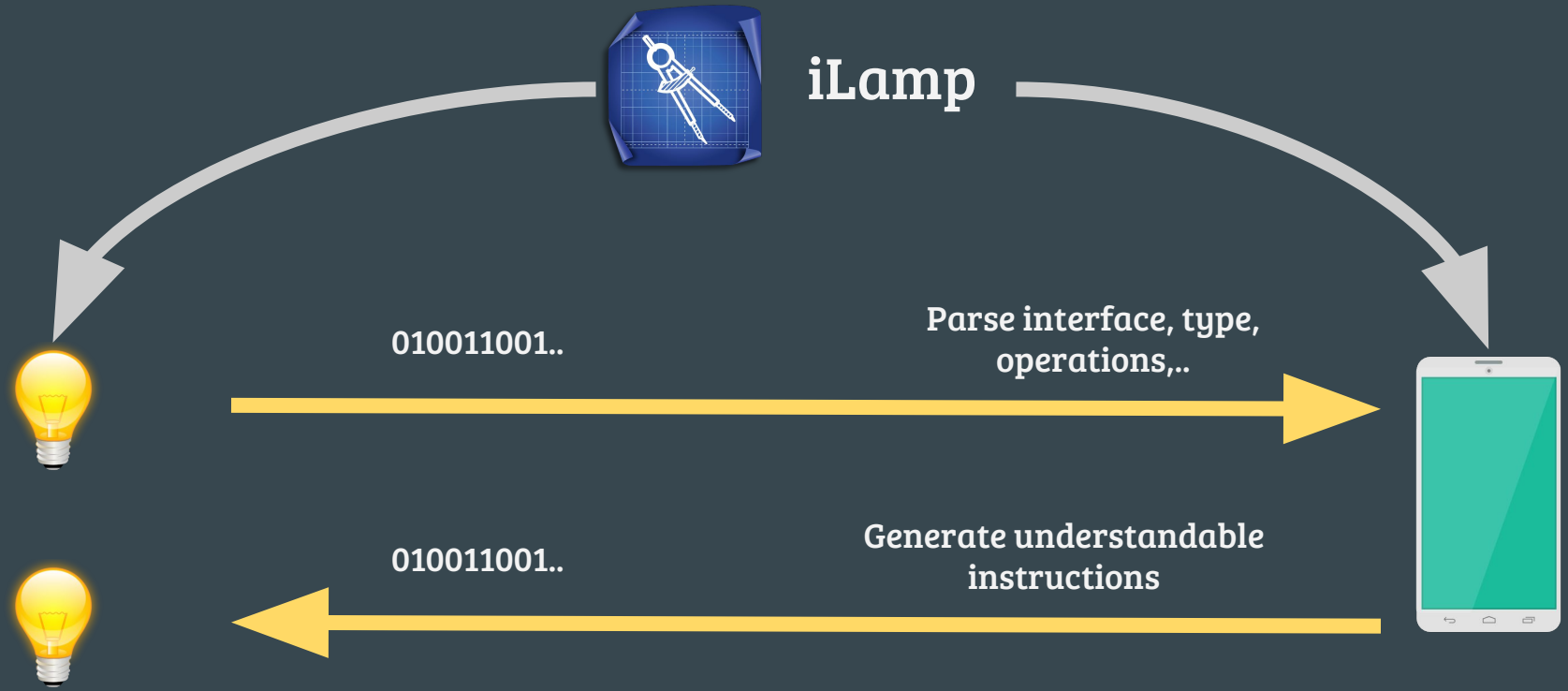
Runtime Architecture

Custom Firmware	NETWORK		
	BT	Eth	..

Custom Firmware	NETWORK
	BT

DSL			
RUNTIME	NETWORK		
	BT	Eth	..

Communication between devices



The Application Logic

...

Software Properties

Turn on lights in vicinity

While moving..

Disregard dead batteries in lamps

Turn off lamp when it leaves vicinity

The goal

```
with {l is Lamp | l.isNear() } do
  l.on
when !this.isNear():
  l.off
exception:
  maintenance.alert(l.id(), “Threw exception”);
```

Software Properties

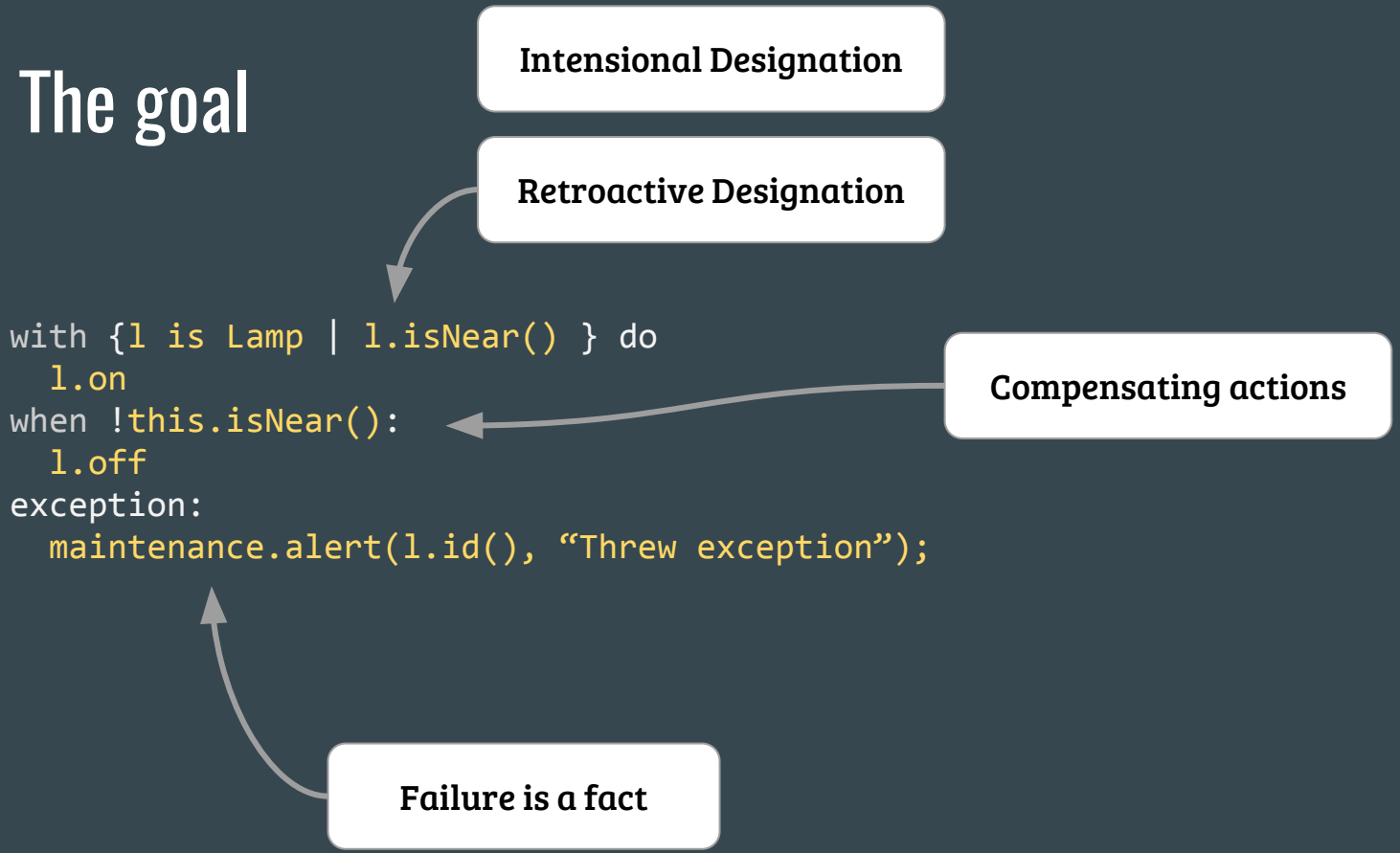
Intensional Designation

Retroactive Designation

Failure

Compensating actions

The goal


Intensional Designation
Retroactive Designation
with {l is Lamp | l.isNear() } do
l.on
when !this.isNear():
l.off
exception:
maintenance.alert(l.id(), “Threw exception”);
Compensating actions
Failure is a fact

Conclusion

Writing Walking lights today is possible but we need to:

- Manage heterogeneous devices
- Implement against very specific API's and capabilities
- Manage the network by hand

We address these issues with

- Abstraction over device interfaces
- Abstraction over discovery and network management
- Abstract over device heartbeats and monitoring