Microservices for the IoT

JavaOne™

luminis
Conversing worlds

luminis
Conversing worlds

luminis
Conversing worlds

Marcel Offermans

Director at Luminis Technologies
Member of the Apache Software Foundation

@m4rr5 marcel.offermans@luminis.eu

Agenda

- Microservices
- Case: Entertainment System
- Build, run and deploy microservices
- Internet of Things
- IoT Deployment
- Wrapup

Microservices

Design

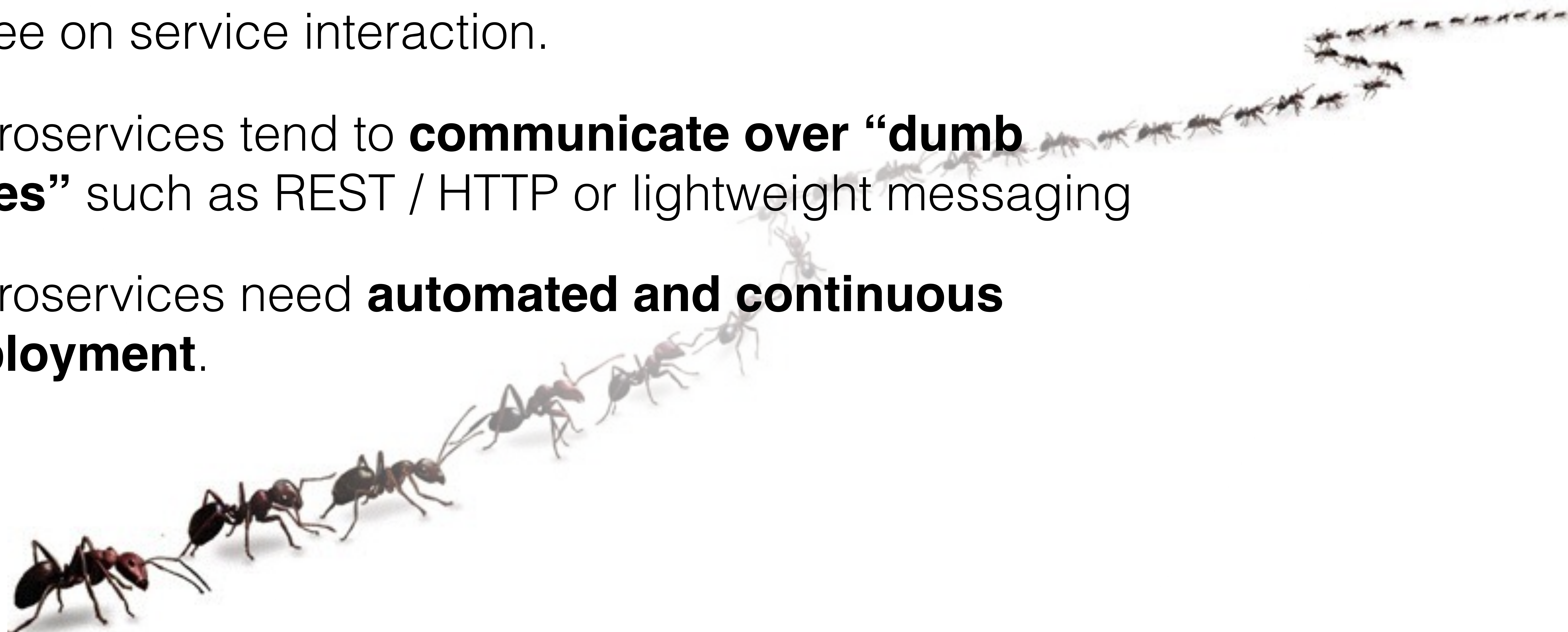
- **Business Capabilities are leading** when splitting an application into microservices.
- **The only constant is Change**. Abstract interfaces. Version them. Consider rate of change, high cohesion, low coupling.
- **Things will fail.** Design for failure and be explicit about how a microservice will deal with and recover from failure.

Anatomy

- Microservices **communicate via services**, which provide a public, versioned contract. Implementations are hidden.
- Microservices **have their own life cycle**, so they can be separately deployed.
- Microservices **own their own data**. This can lead to polyglot persistence and that is fine.
- Microservices **use transaction-less coordination** and are built for eventual consistency.

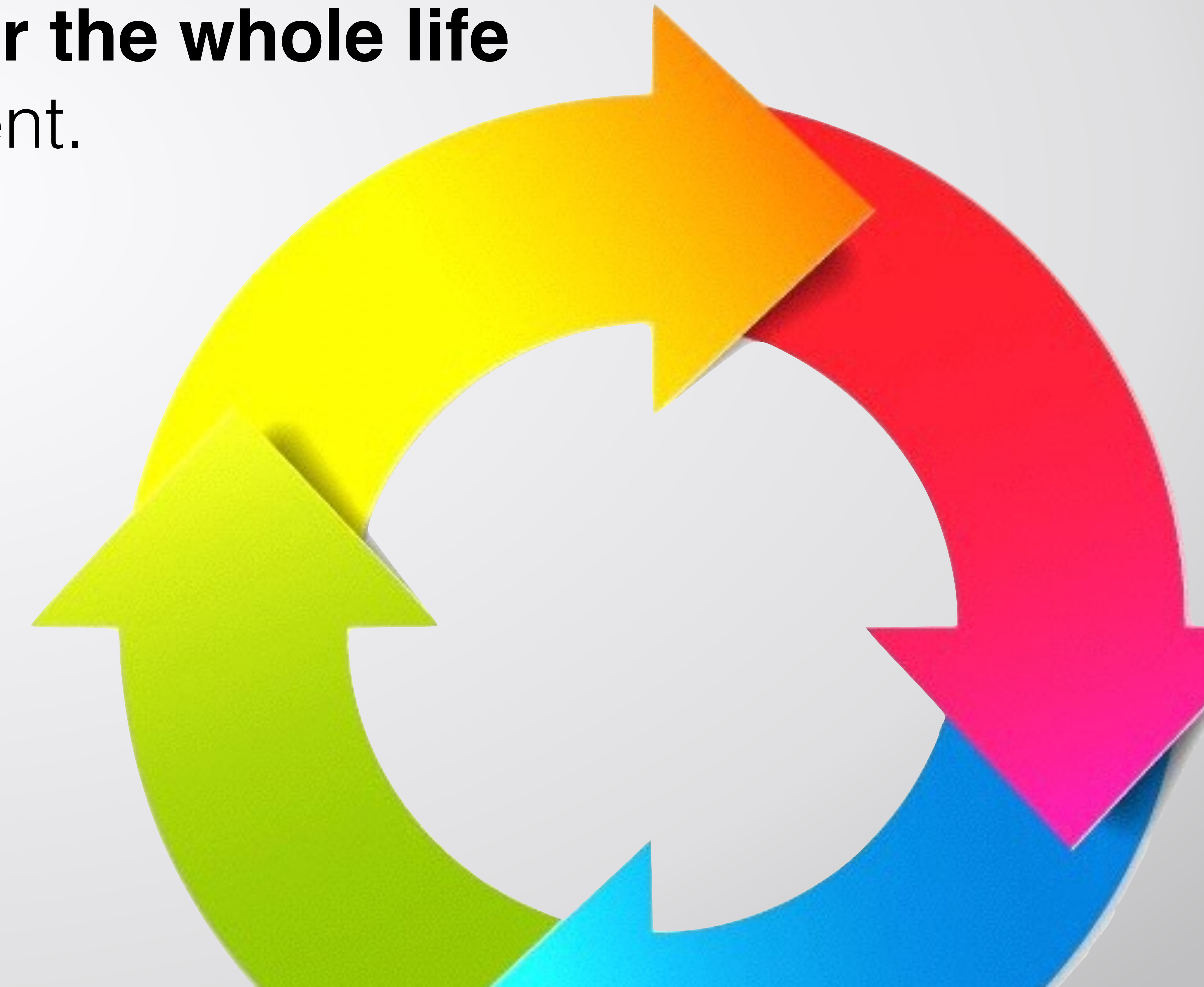
Orchestration

- Microservices offer strong decoupling, leaving us free to **choose implementation languages**.
- There is **less need for formal standards**, as long as you agree on service interaction.
- Microservices tend to **communicate over "dumb pipes"** such as REST / HTTP or lightweight messaging
- Microservices need **automated and continuous deployment**.



Process

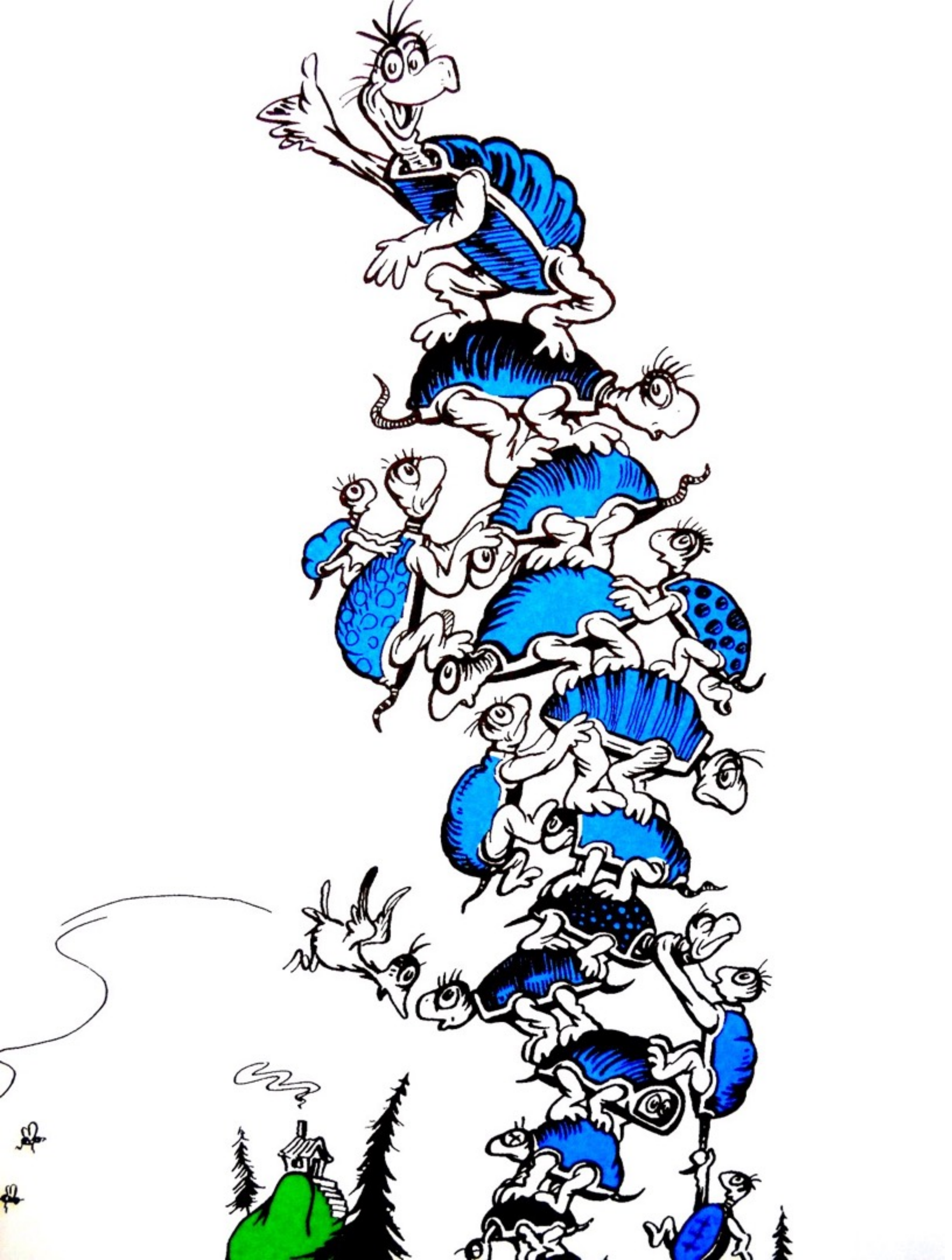
- Make the **team responsible for the whole life cycle** of a product or component.
- Remember **Conway's Law**!



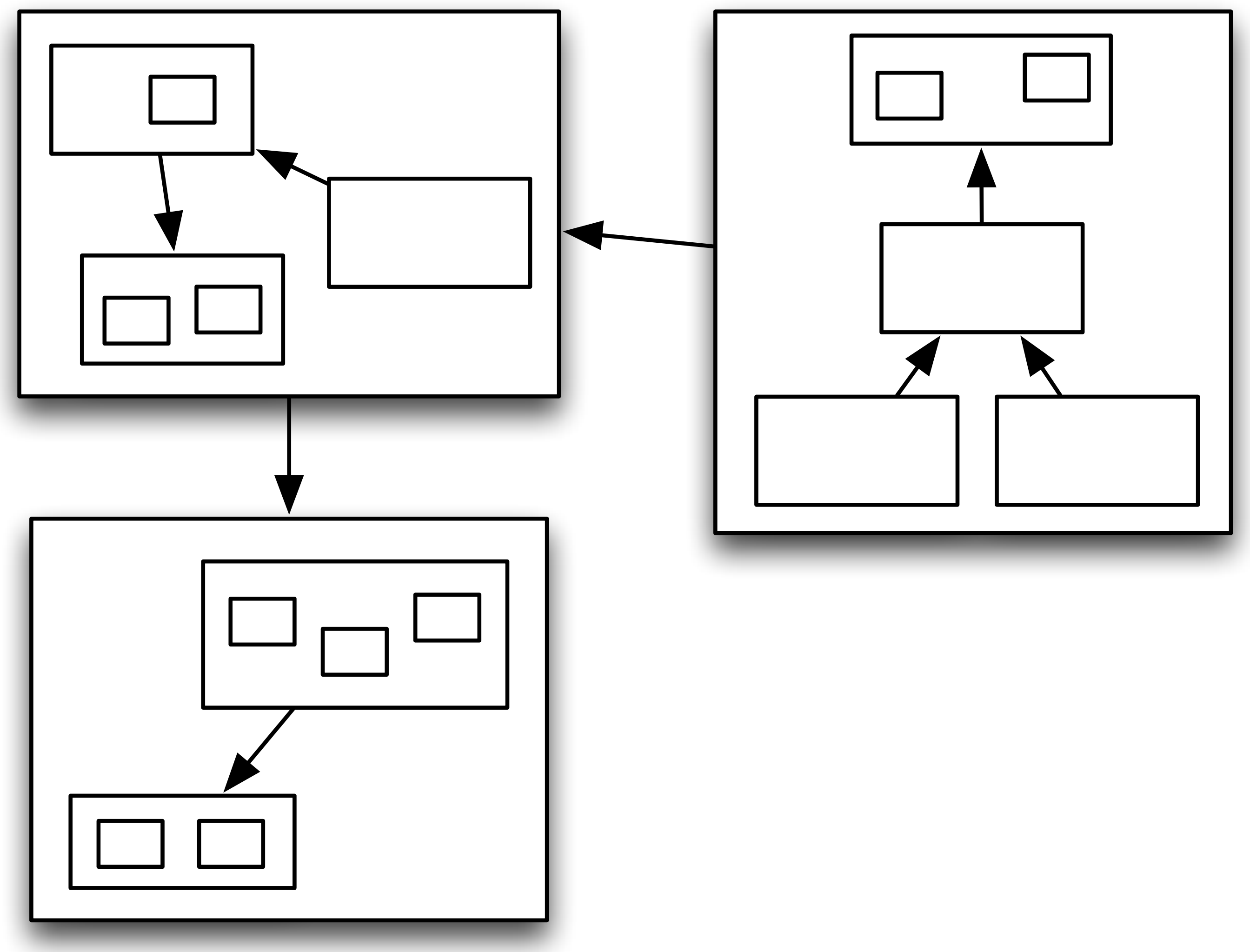
Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure.

Melvyn Conway, 1967

The case for modularity



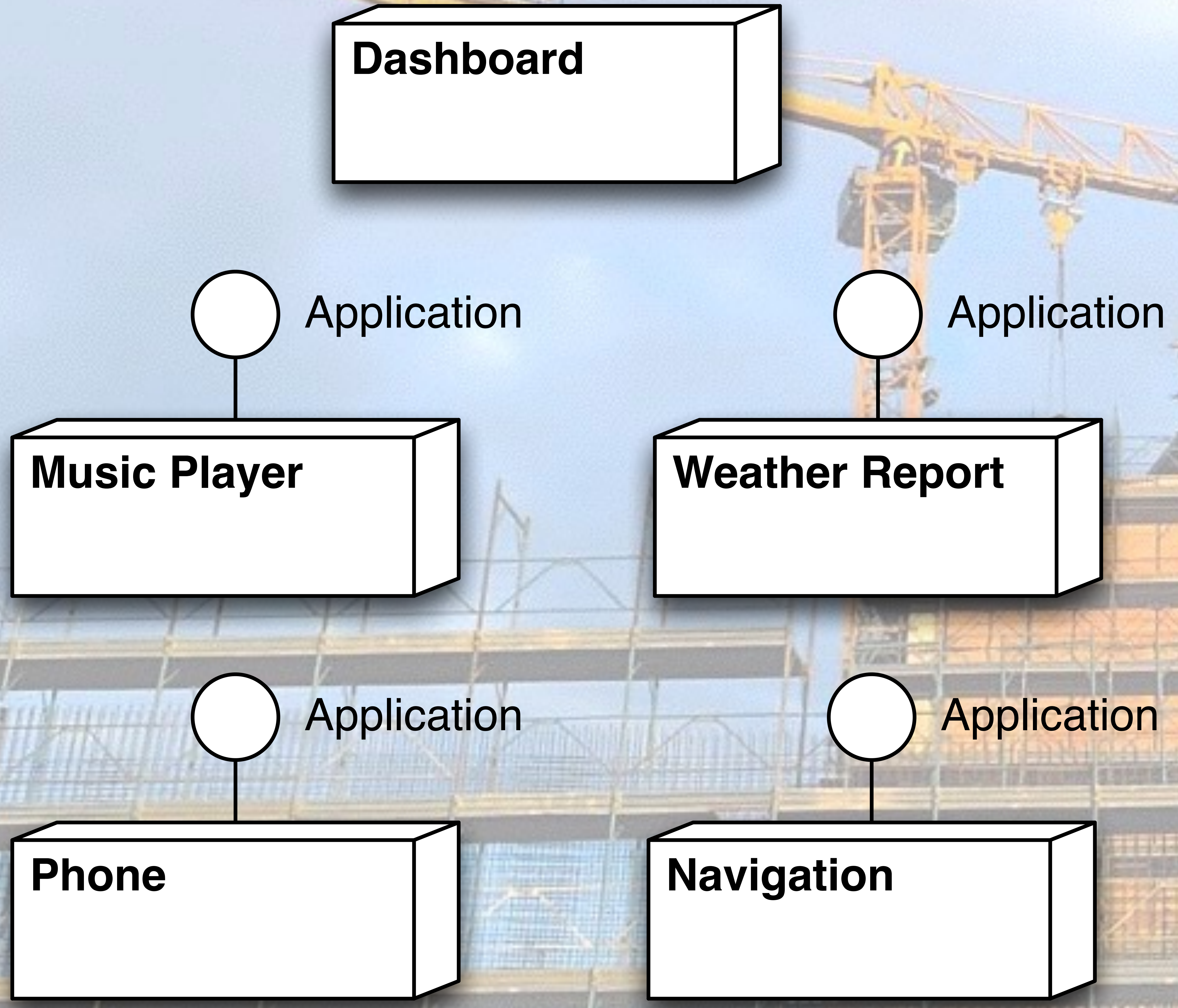
At all levels

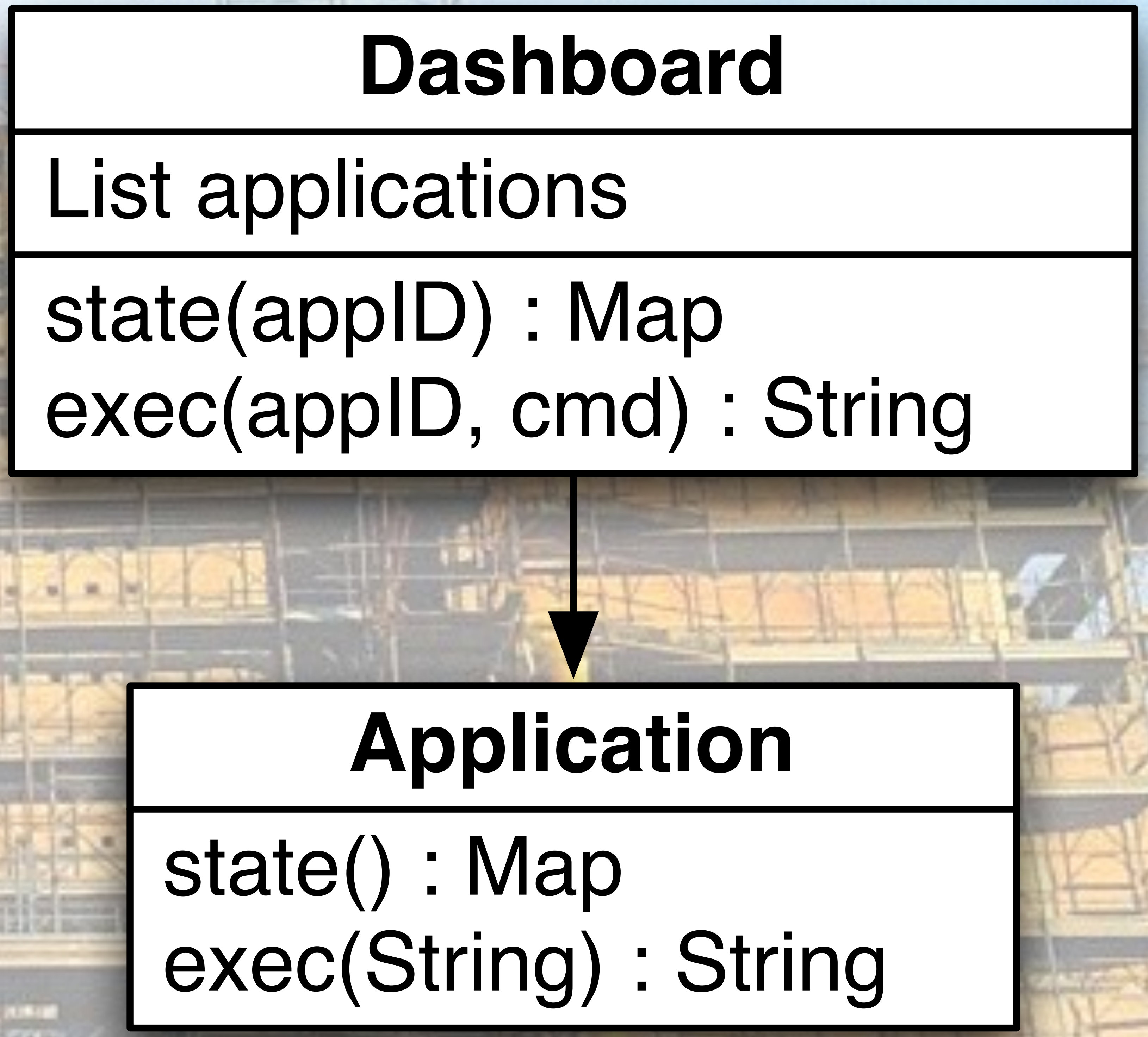


"it's turtles all the way down"

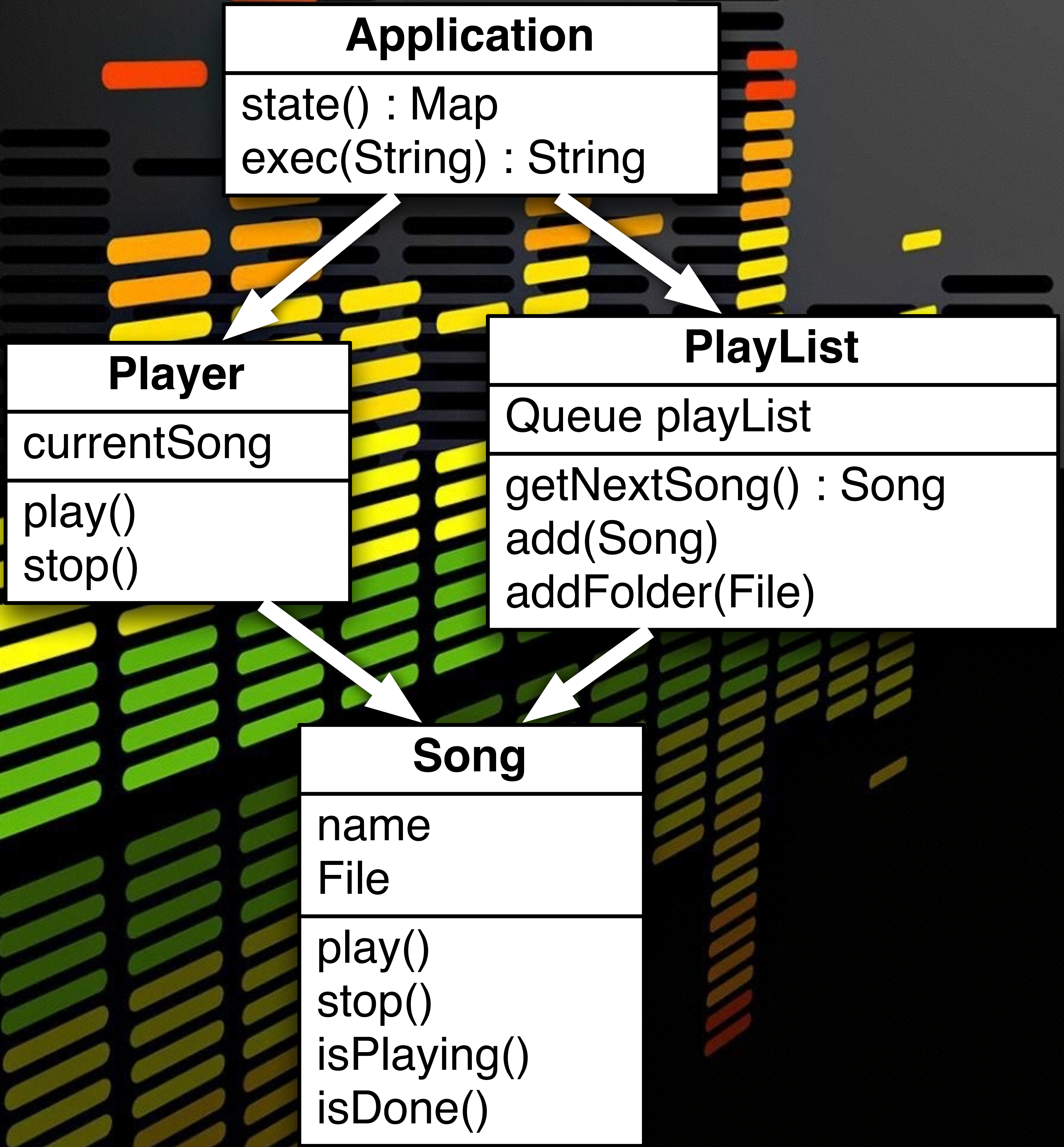
Case: Entertainment System

Architecture





Music Player



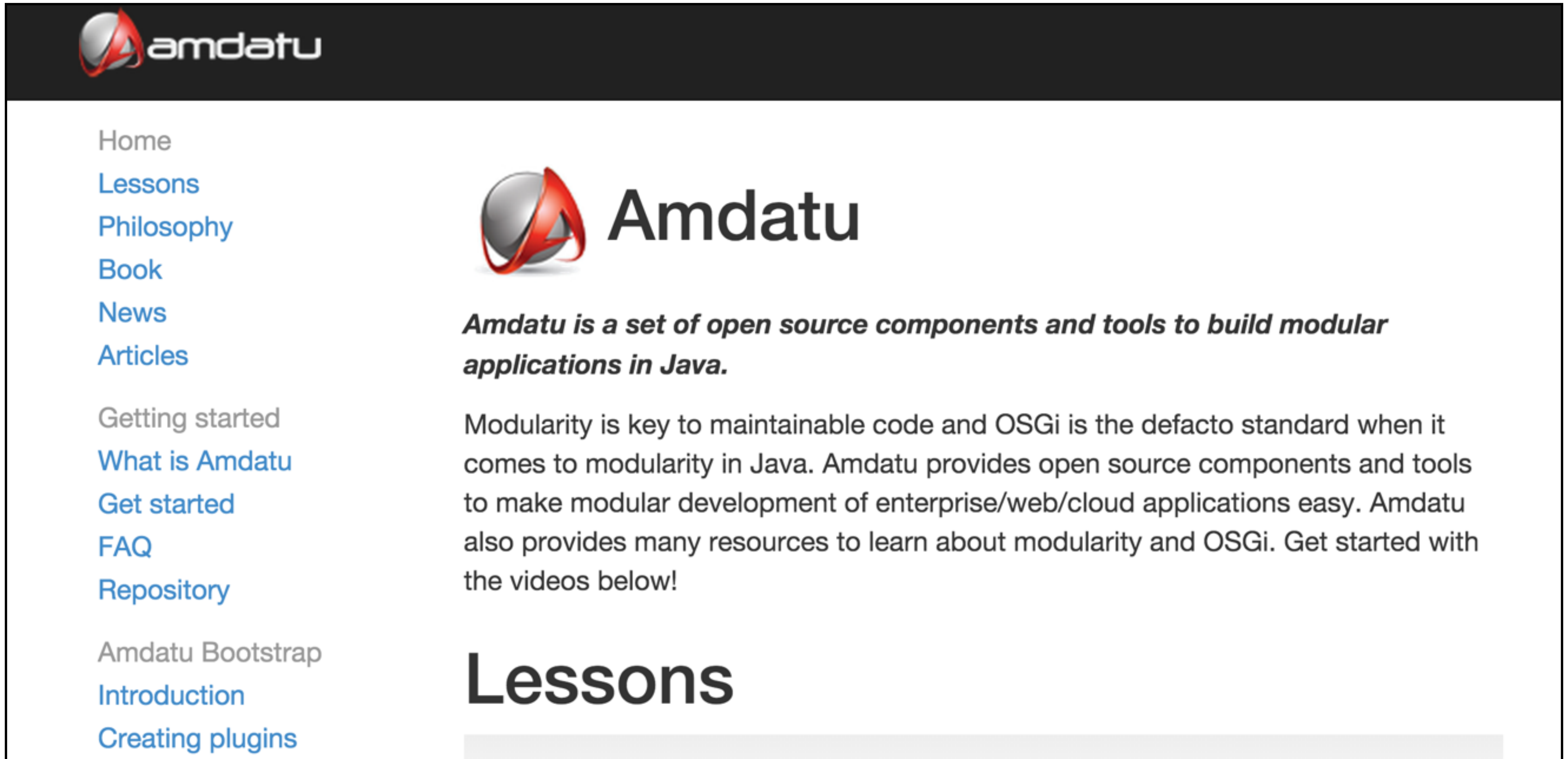
Building a microservice

What is OSGi?

- Provides a container where modules can be easily deployed and versioned
- Within a module, hides implementation details and allows explicit, versioned sharing of code
- Provides a service registry that allows modules to publish and consume services to interact
- It’s the de-facto module system for Java: proven technology, works on all Java versions, usable from embedded to enterprise

What is Amdatu?

Amdatu is a set of open source components and tools to build modular applications in Java.



Home
Lessons
Philosophy
Book
News
Articles

Getting started
What is Amdatu
Get started
FAQ
Repository

Amdatu Bootstrap
Introduction
Creating plugins

Amdatu

Amdatu is a set of open source components and tools to build modular applications in Java.

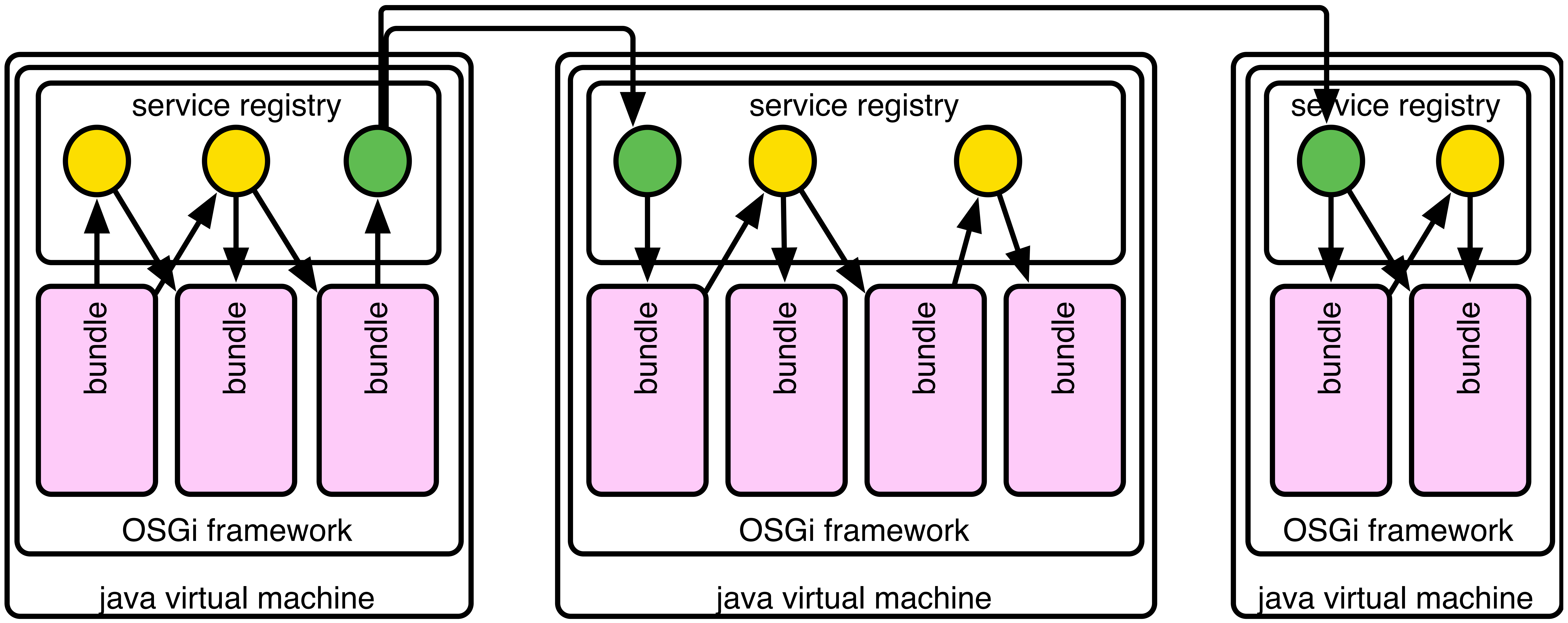
Modularity is key to maintainable code and OSGi is the defacto standard when it comes to modularity in Java. Amdatu provides open source components and tools to make modular development of enterprise/web/cloud applications easy. Amdatu also provides many resources to learn about modularity and OSGi. Get started with the videos below!

Lessons

demo

Running microservices

Remote Services



demo

Continuous Deployment

Apache ACE is a software distribution framework that allows you to centrally manage and distribute software modules to target systems.

Continuous Deployment:

1. Checkout and build the code
2. Upload modules to Apache ACE
3. Group them into microservices
4. Assign microservices to distributions
5. Assign distributions to target systems

demo

Internet of Things

“Things”

- ✳ **Machines**
- ✳ **Appliances**
- ✳ **Vehicles**
- ✳ **Buildings**
- ✳ **Plants**
- ✳ **Humans**
- ✳ **Animals**

Identification

- Important to uniquely identify each thing
- IPv6 provides uniqueness:
  “enough addresses for every atom on earth… times 100”

Sensors

- ✳ **Position**
- ✳ **Temperature**
- ✳ **Motion**
- ✳ **RFID**
- ✳ **Sight**
- ✳ **Sound**

Communication

Bluetooth®

ZigBee®

WiFi™

Autonomous

Challenges

- ✳ More data to process and store
- ✳ Huge security concerns
- ✳ Privacy?
- ✳ Internet of things will change the world

IoT Deployment

Deploying microservices to IoT devices

- all microservices are installed on the same device
- the device is typically resource constrained
- software updates have to be done over limited connections (speed/bandwidth)

demo

Wrapping up

We have…

- explored microservices and modularity
- built a microservices Java application using OSGi and Amdatu
- packaged, deployed and ran this application in different ways
- seen how we can reduce the footprint of the application for an IoT device

Provisioning Server
http://ace.apache.org/

Apache ACE

Cloud OSGi services
http://www.amdatu.org/

amdatu

Eclipse OSGi plugin
http://bndtools.org/

bndtools

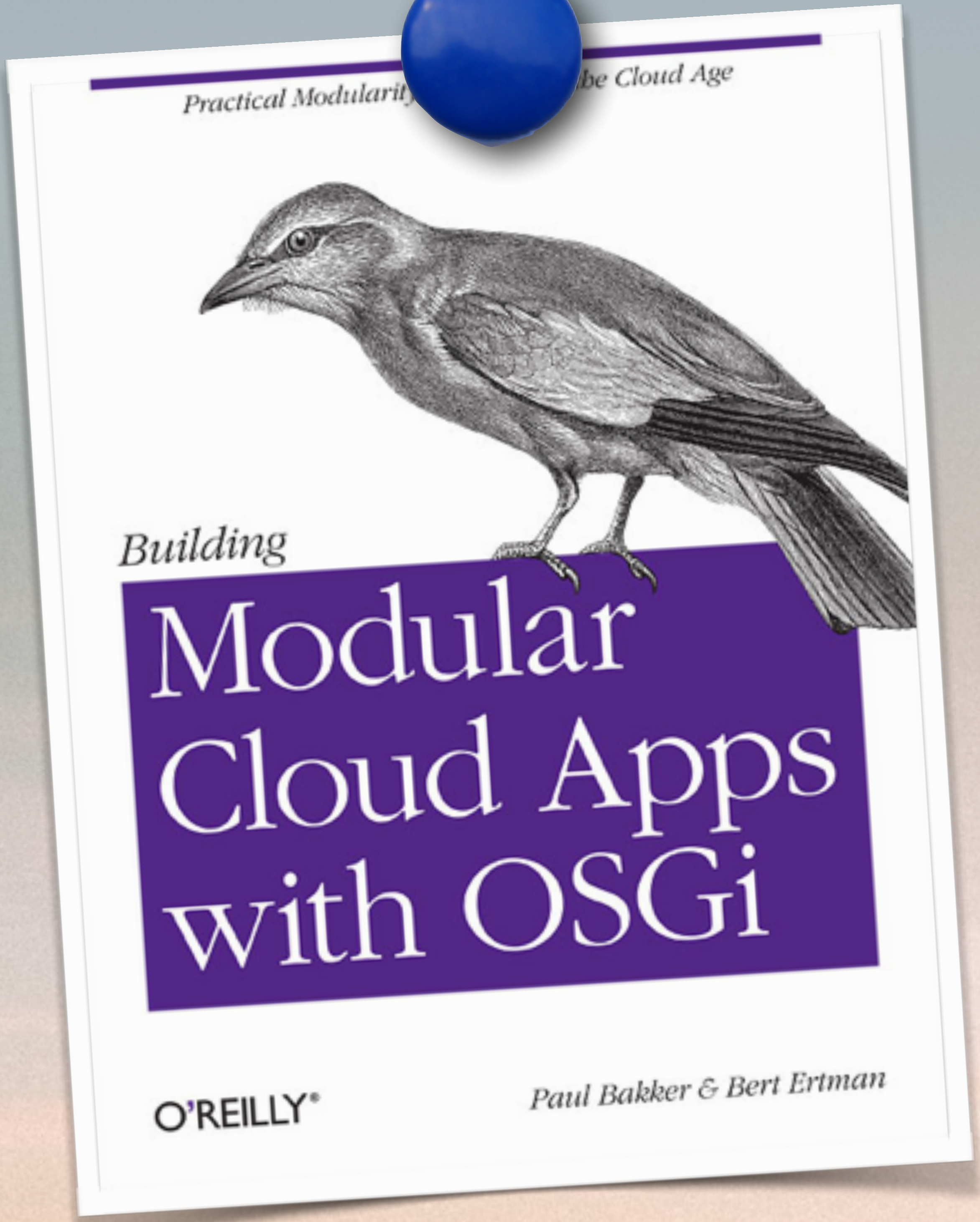
Coming soon:

CLOUD RTI

That's us
http://luminis.eu/

luminis
Conversing worlds

Demo code
https://bitbucket.org/marrs/microservices-for-the-iot/


Practical Modularity ... the Cloud Age

Building
Modular Cloud Apps with OSGi

O'REILLY

Paul Bakker & Bert Ertman

Zero-Downtime Java Deployments with Docker and Kubernetes [CON2608]
Paul Bakker, Arjan Schaaf
Thursday, Oct 29, 4:00 p.m. | Parc 55—Market Street

Thanks!

Microservices for Mortals [CON2488]
Bert Ertman
Wednesday, Oct 28, 8:30 a.m. | Parc 55—Powell I/II
Wednesday, Oct 28, 1:00 p.m. | Parc 55—Cyril Magnin I

Enjoy JavaOne!