Introduction to Microservices

Memi Lavi
www.memilavi.com

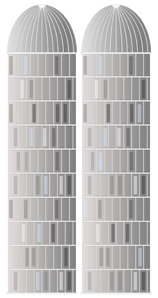
Microservices are a result of problems with two architecture paradigms:

- Monolith
- SOA

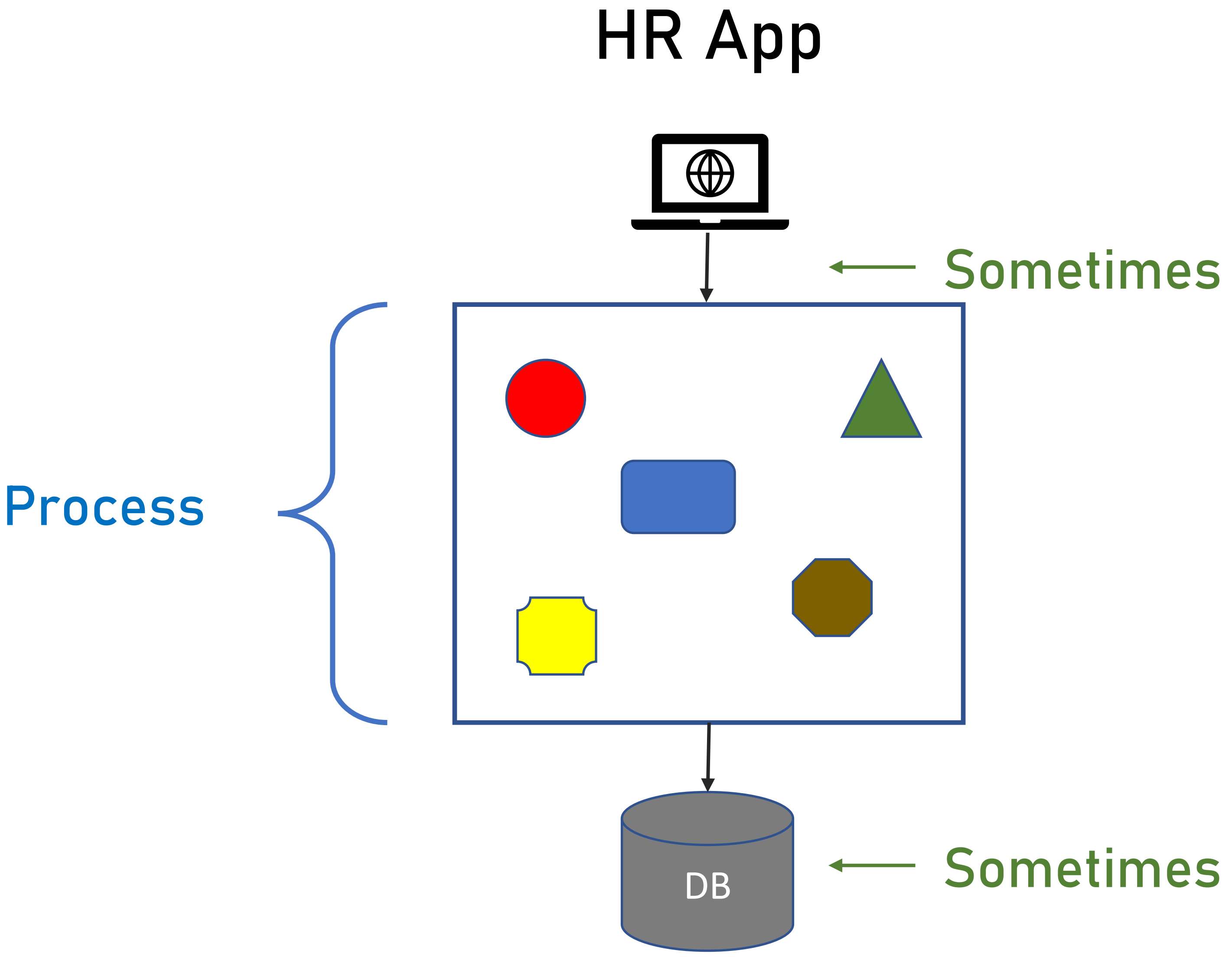
Monolith

Monolith Architecture

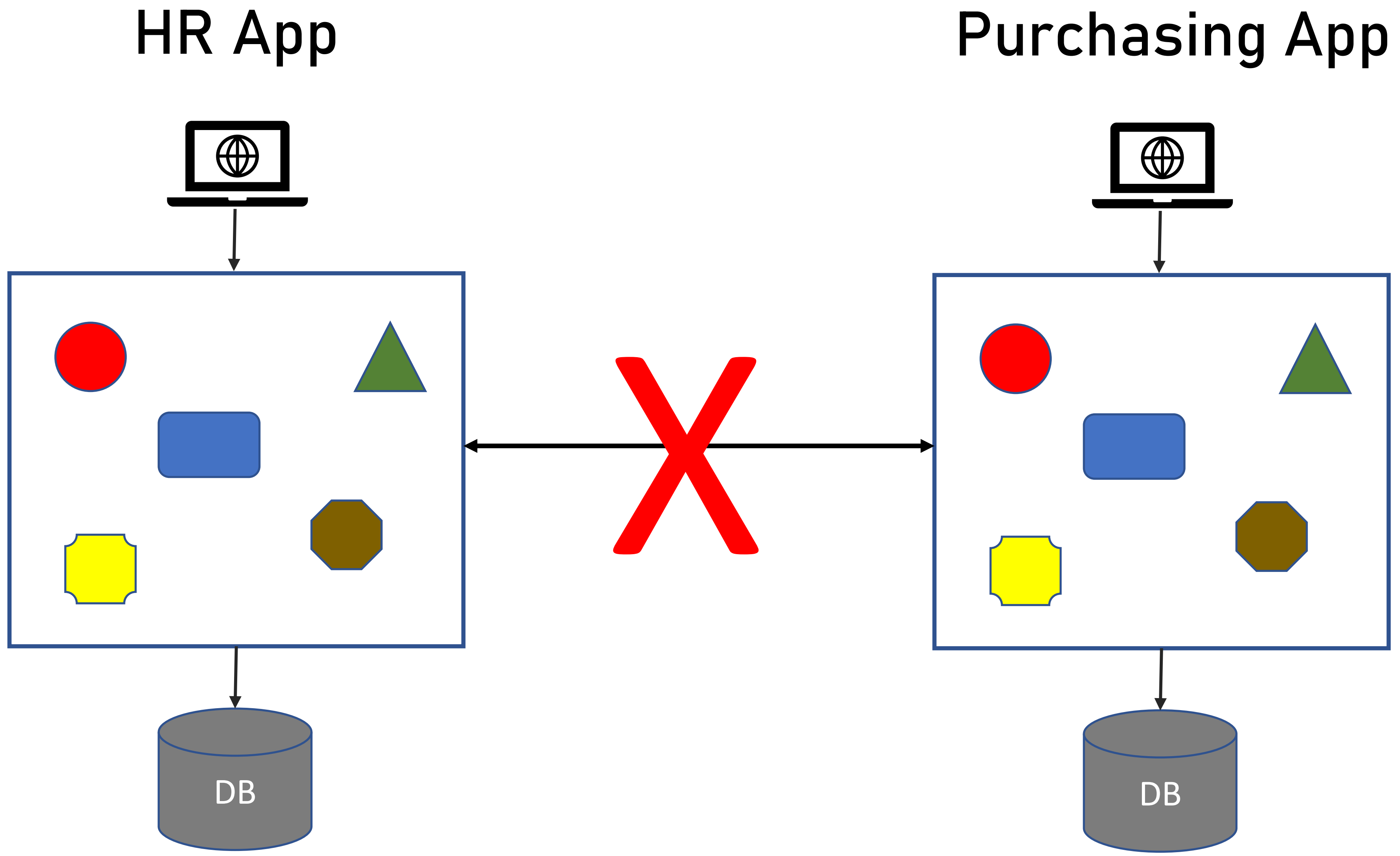
- The original architecture
- All software components are executed in a single process
- No distribution of any kind
- Strong coupling between all classes
- Usually implemented as Silo



Monolith Architecture - Example

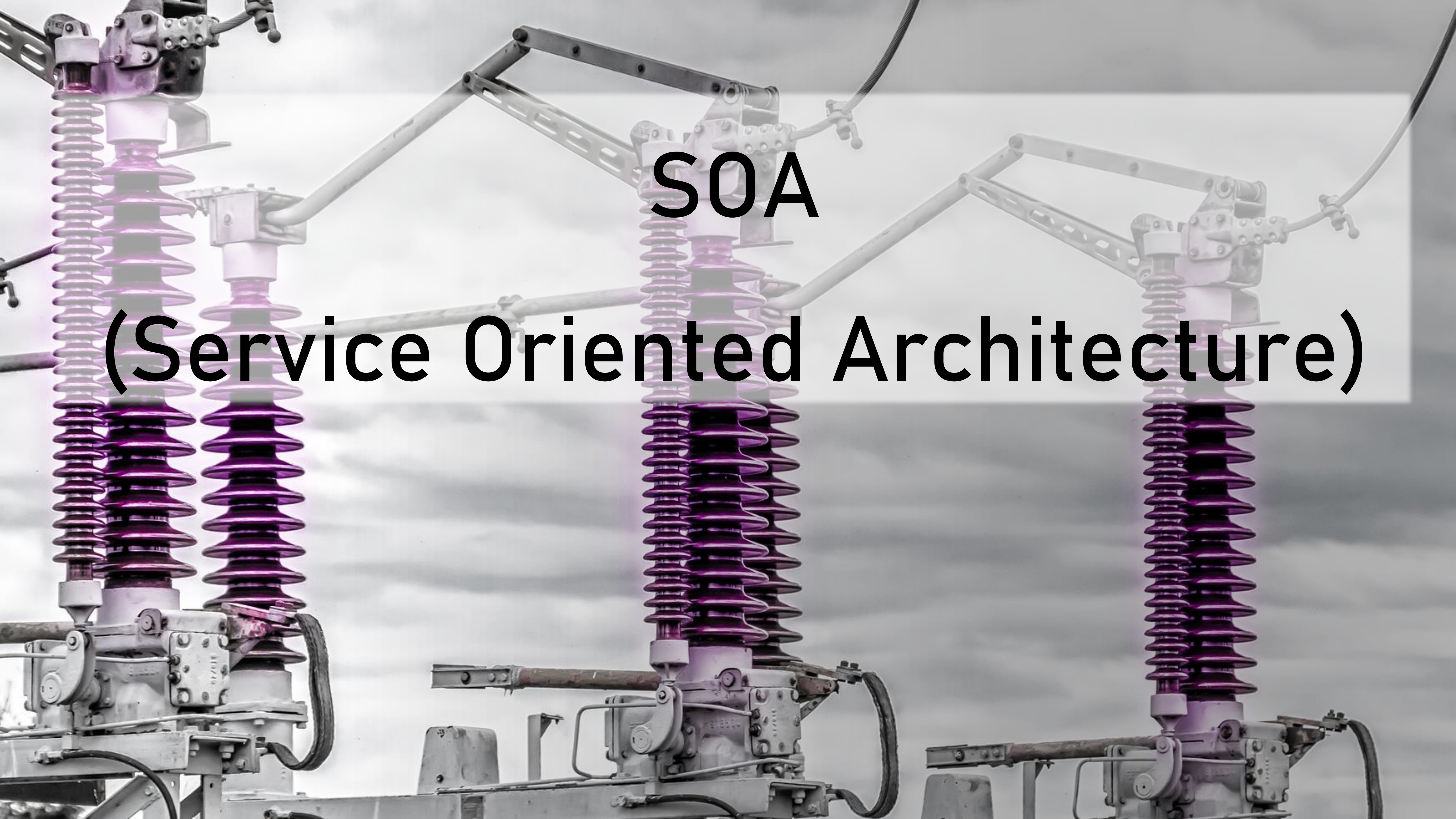


Monolith Architecture – Example



Monolith Architecture Pros

- Easier to design
- Performance



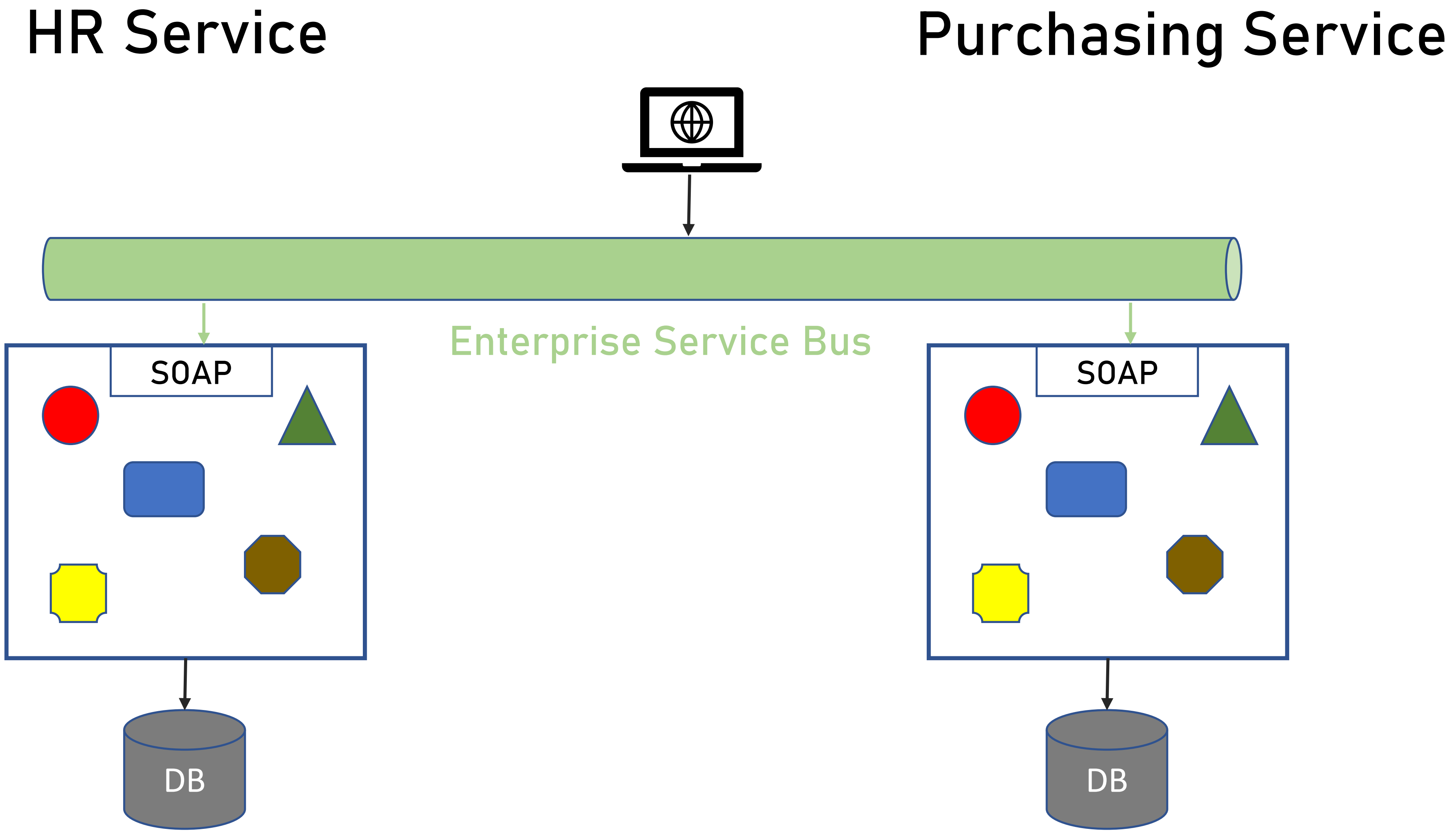
SOA

(Service Oriented Architecture)

Service Oriented Architecture

- First coined in 1998
- Apps are services exposing functionality to the outside world
- Services expose metadata to declare their functionality
- Usually implemented using SOAP & WSDL
- Usually implemented with ESB

SOA Architecture - Example



SOA Pros

- Sharing Data & Functionality
- Polyglot Between Services

Problems

- A lot of problems were found in both paradigms
- Problems relevant to technology, deployment, cost and more
- We'll begin with the Monolith and then move on to SOA

Single Technology Platform

- With monolith, all the components must be developed using the same development platform
- Not always the best for the task
- Can’t use specific platform for specific features
- Future upgrade is a problem – need to upgrade the whole app

Inflexible Deployment

- With monolith, new deployment is always for the whole app
- No way to deploy only part of the app
- Even when updating only one component – the whole codebase is deployed
- Forces rigorous testing for every deployment
- Forces long development cycles

Inefficient Compute Resources

- With monolith, compute resources (CPU and RAM) are divided across all components
- If a specific component needs more resources – no way to do that
- Very inefficient

Large and Complex

- With monolith, the codebase is large and complex
- Every little change can affect other components
- Testing not always detects all the bugs
- Very difficult to maintain
- Might make the system obsolete

Complicated and Expensive ESB

- With SOA, the ESB is one of the main components
- Can quickly become bloated and expensive
- Tries to do everything
- Very difficult to maintain

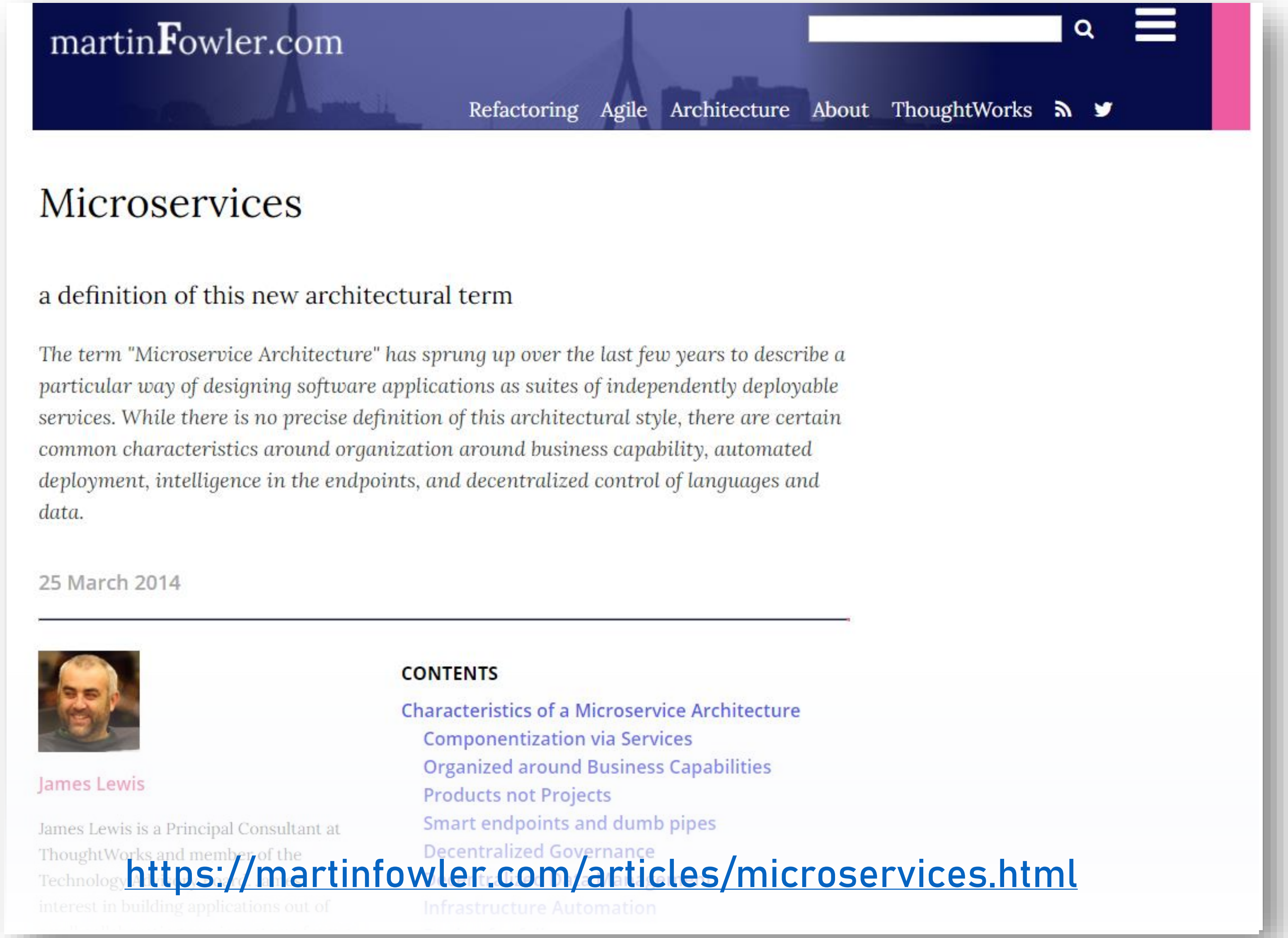
Lack of Tooling

- For SOA to be effective, short development cycles were needed
- Allow for quick testing and deployment
- No tooling existed to support this
- No time saving was achieved

Microservices to the rescue

- Problems with monolith and SOA led to a new paradigm
- Has to be modular, with simple API
- The term “microservices” first appeared in 2011
- In 2014 Martin Fowler and James Lewis published their “Microservices” article
    - Became the de-facto standard for Microservices definition

The Article


martinFowler.com

Refactoring Agile Architecture About ThoughtWorks

Microservices

a definition of this new architectural term

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

25 March 2014

James Lewis

James Lewis is a Principal Consultant at ThoughtWorks and member of the Technology Advisory Board. James' interest in building applications out of

CONTENTS

https://martinfowler.com/articles/microservices.html

Characteristics of Microservices

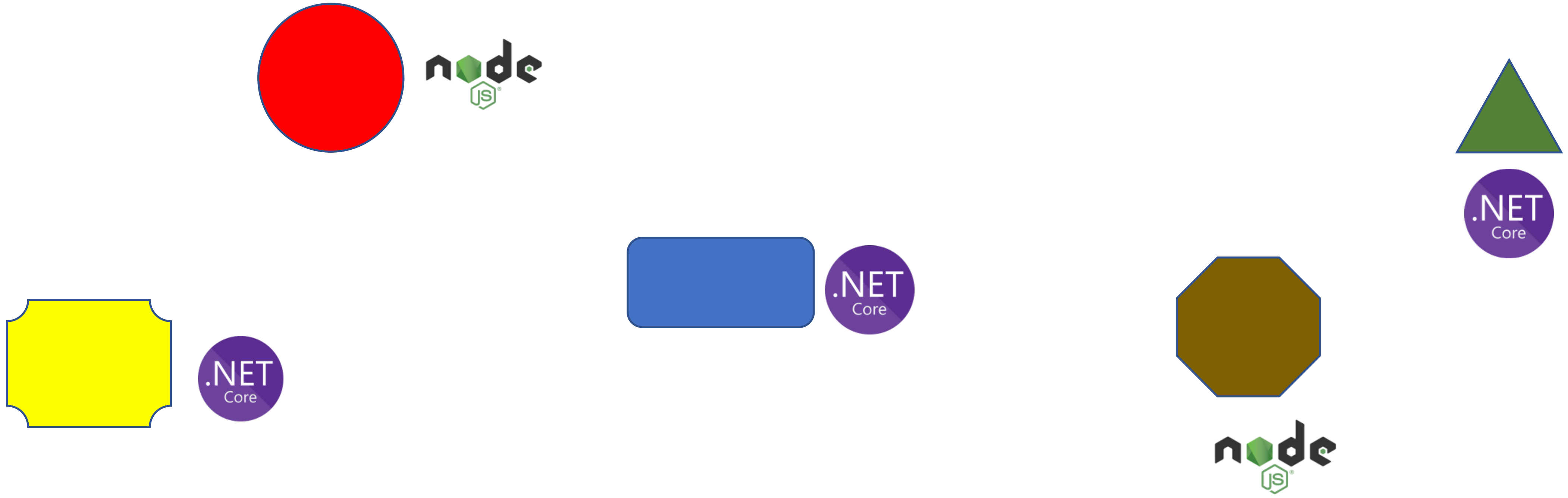
- Componentization via Services
- Organized Around Business Capabilities
- Products not Projects
- Smart Endpoints and Dumb Pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for Failure
- Evolutionary Design

Problems Solved

- We discussed problems caused by Monolith and SOA
- Microservices solve these problems
- Let’s see how…

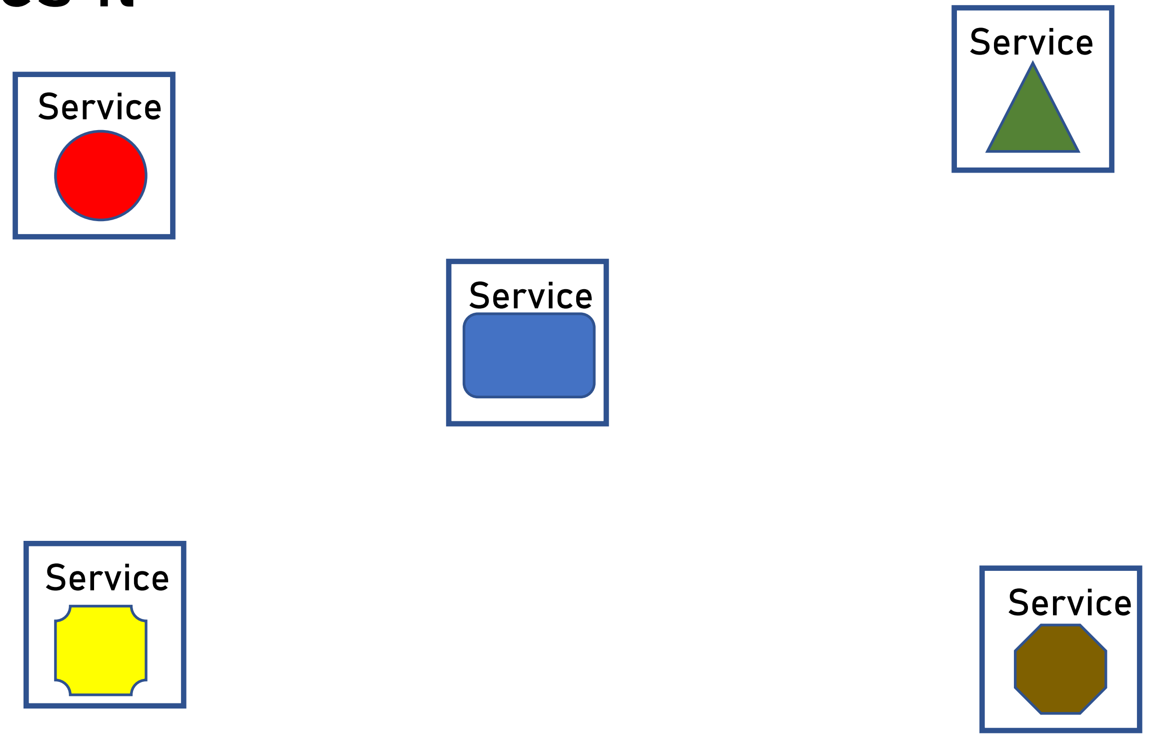
Single Technology Platform

- With Microservices, the Decentralized Governance attribute solves it



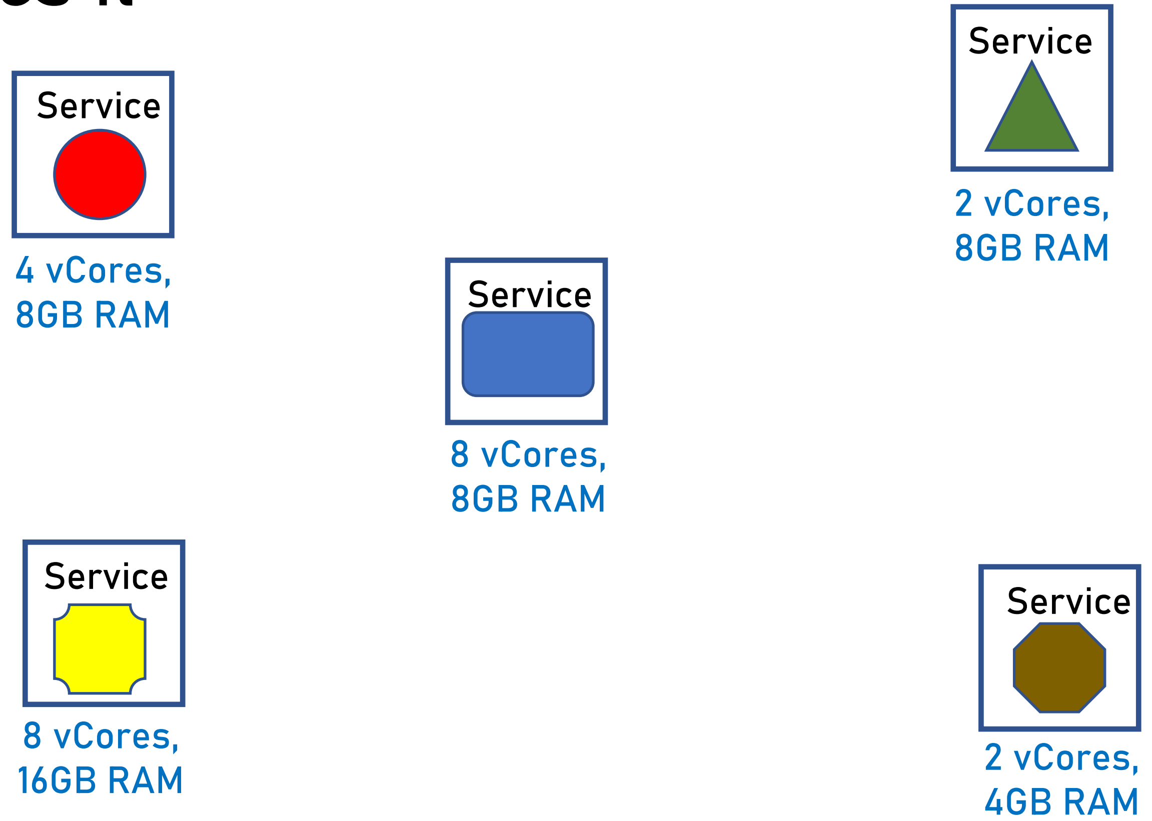
Inflexible Deployment

- With Microservices, the Componentization via Services attribute solves it



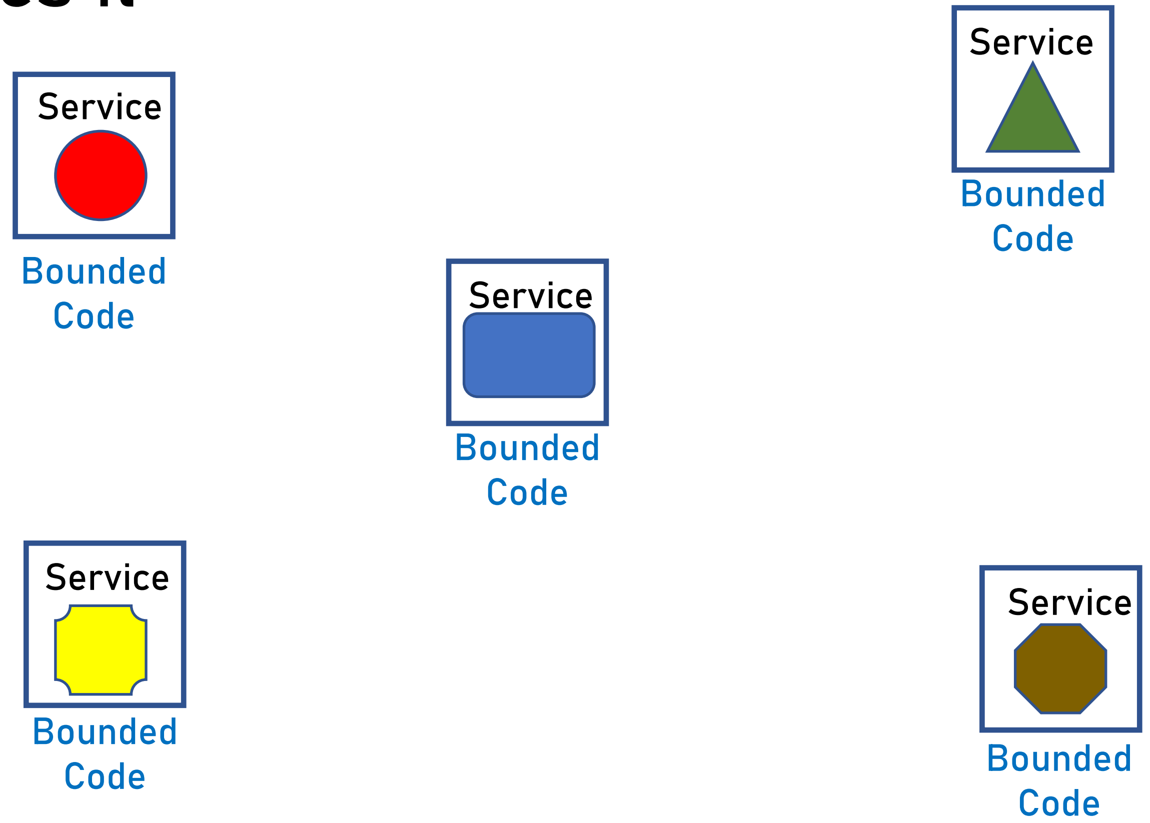
Inefficient Compute Resources

- With Microservices, the Componentization via Services attribute solves it



Large and Complex

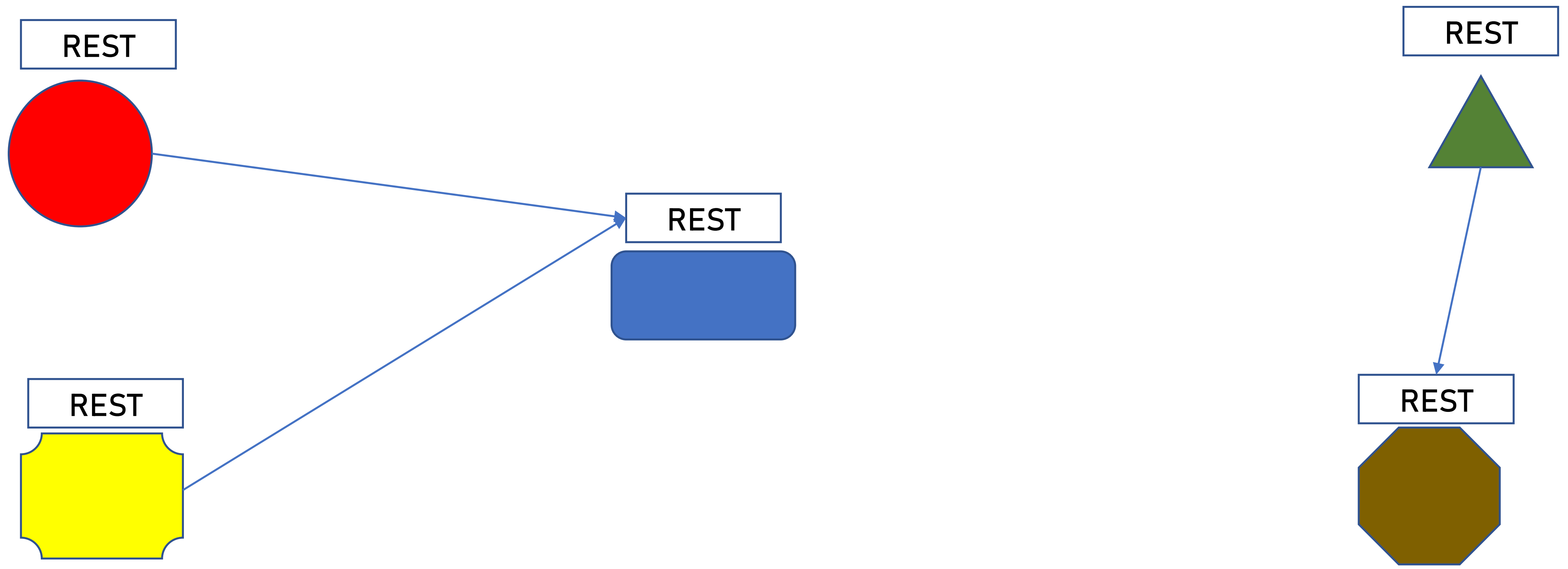
- With Microservices, the Componentization via Services attribute solves it



Complicated and Expensive ESB

- With Microservices, the Smart Endpoint and Dumb Pipes

  attribute solves it



Lack of Tooling

- With Microservices, the Infrastructure Automation attribute solves it
- Automates testing and deployment
- Provides short deployment cycles
- Make the architecture efficient and effective