The Jolie Programming Language

Reducing the distance between service models and code

Fabrizio Montesi
University of Southern Denmark (SDU)

twitter.com/famontesi

fabrizio#1961

@ fmontesi@imada.sdu.dk

SDU

DEPARTMENT OF MATHEMATICS
AND COMPUTER SCIENCE

Theory

Practice

Theory

Practice

What is a (micro)service?

An autonomous software application that interacts with other applications by means of message passing.

What is Jolie?

- A service-oriented programming language.
- A collaborative project.
  - Open source.
  - Adopted for microservices, system integration, and web applications.
  - Active collaboration within the Microservices Community.

https://microservices.community

- Website: https://jolie-lang.org

In collaboration with Microservices Community

Get in touch!

- Chat with us on DISCORD : https://discord.gg/yQRTMNX

- GitHub: https://github.com/jolie/jolie

- Twitter: https://twitter.com/jolielang

- Mailing list: jolie-devel@googlegroups.com

Why Jolie?

And what does service-orientation mean?

Coding

Coding

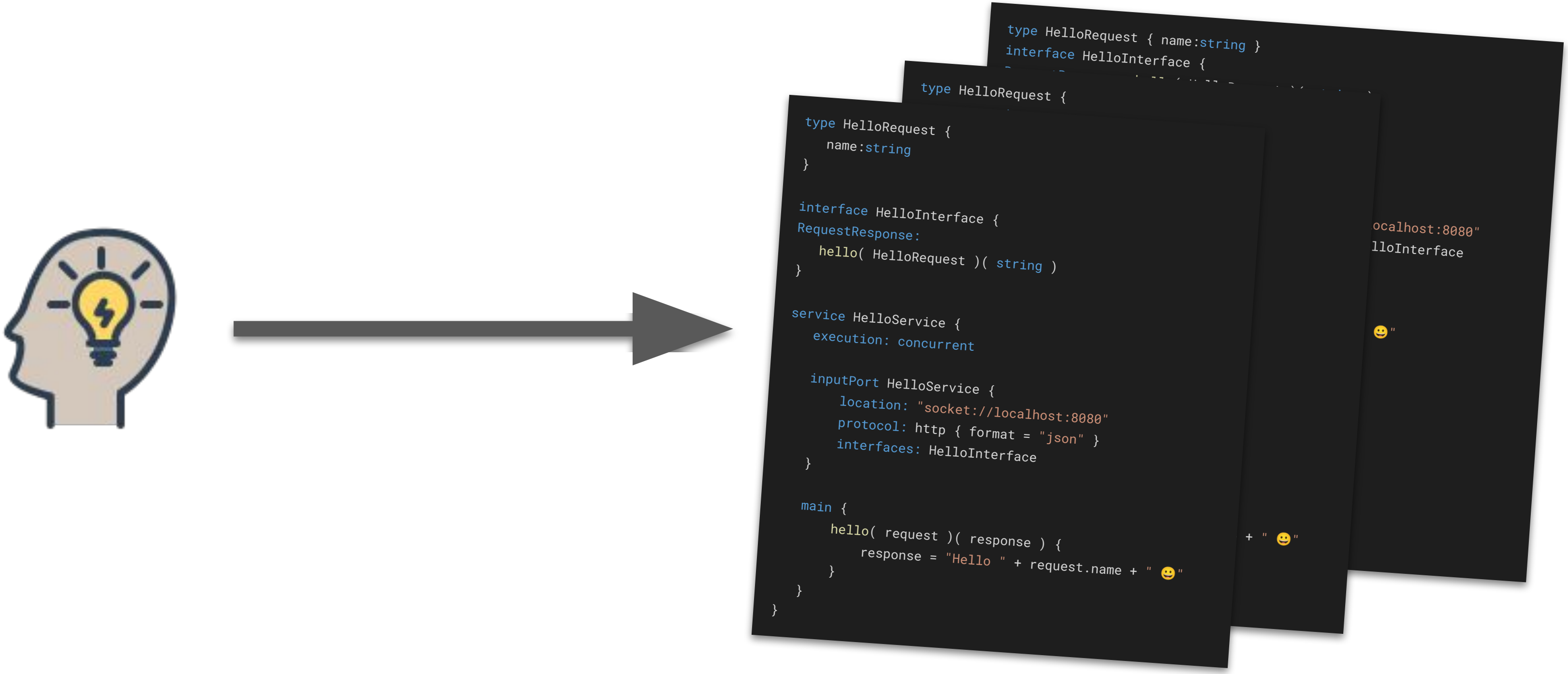
Objective:

Coding

Objective:

Coding

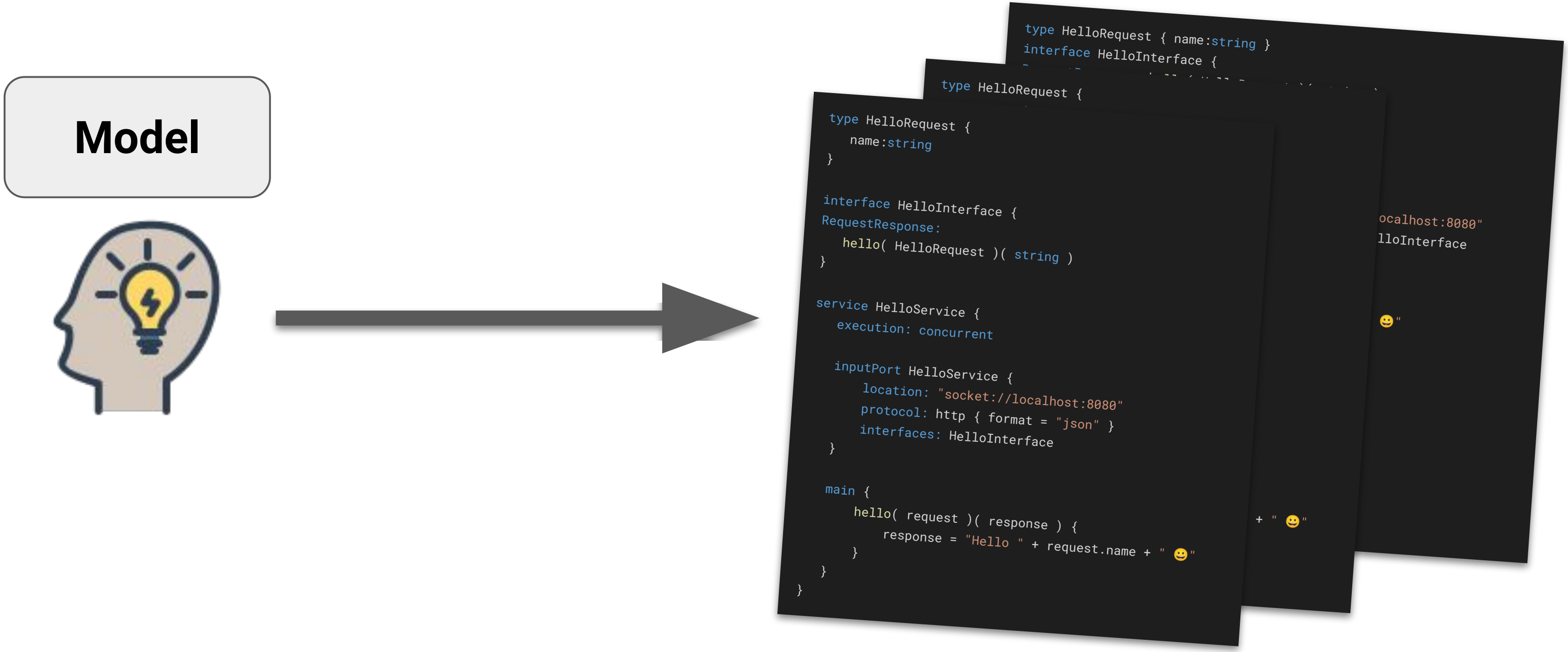
Objective:



```
type HelloRequest {
  name:string
}

interface HelloInterface {
RequestResponse:
  hello( HelloRequest )( string )
}

service HelloService {
  execution: concurrent

  inputPort HelloService {
    location: "socket://localhost:8080"
    protocol: http { format = "json" }
    interfaces: HelloInterface
  }

  main {
    hello( request )( response ) {
      response = "Hello " + request.name + " 😀"
    }
  }
}
```

Coding

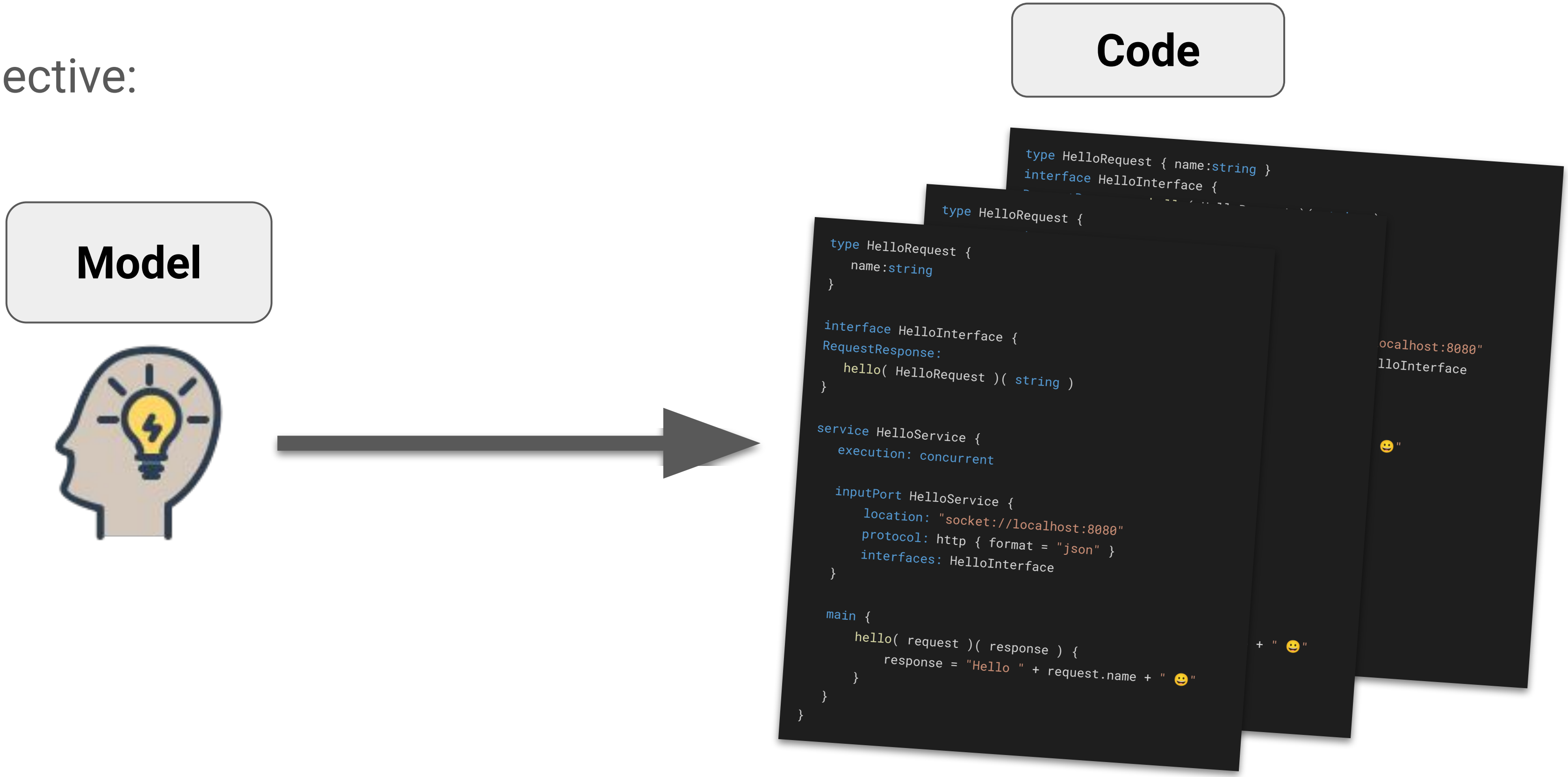
Objective:



```
type HelloRequest {
  name:string
}

interface HelloInterface {
RequestResponse:
  hello( HelloRequest )( string )
}

service HelloService {
  execution: concurrent

  inputPort HelloService {
    location: "socket://localhost:8080"
    protocol: http { format = "json" }
    interfaces: HelloInterface
  }

  main {
    hello( request )( response ) {
      response = "Hello " + request.name + " 😀"
    }
  }
}
```

Coding

Objective:



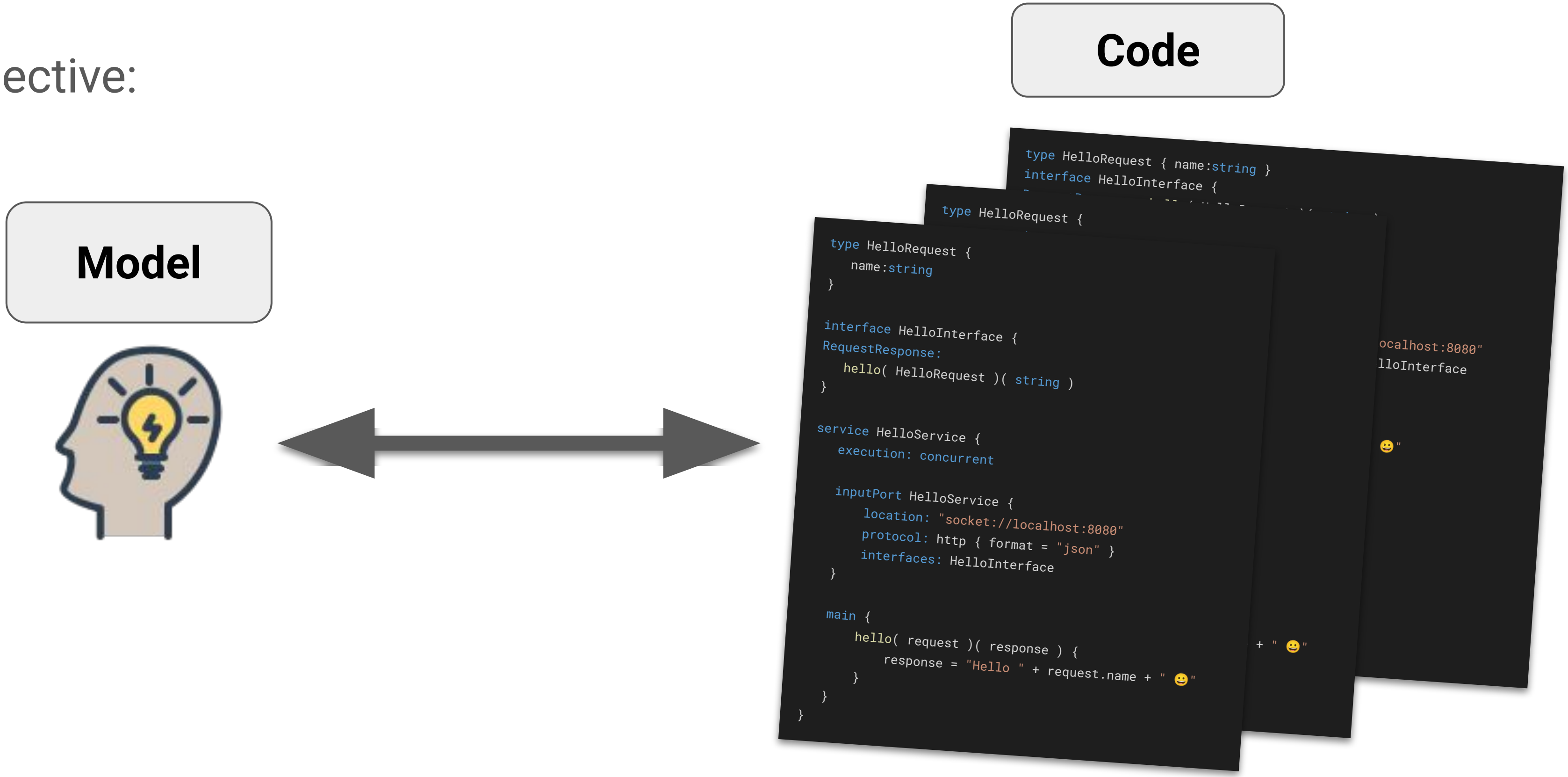
```
type HelloRequest { name:string }
interface HelloInterface {
```

```
type HelloRequest {
```

```
type HelloRequest {
  name:string
}

interface HelloInterface {
RequestResponse:
  hello( HelloRequest )( string )
}

service HelloService {
  execution: concurrent

  inputPort HelloService {
    location: "socket://localhost:8080"
    protocol: http { format = "json" }
    interfaces: HelloInterface
  }

  main {
    hello( request )( response ) {
      response = "Hello " + request.name + " 😀"
    }
  }
}
```

Coding

Objective:



Model

Code

```
type HelloRequest {
  name:string
}

interface HelloInterface {
RequestResponse:
  hello( HelloRequest )( string )
}

service HelloService {
  execution: concurrent

  inputPort HelloService {
    location: "socket://localhost:8080"
    protocol: http { format = "json" }
    interfaces: HelloInterface
  }

  main {
    hello( request )( response ) {
      response = "Hello " + request.name + " 😀"
    }
  }
}
```

Linguistic Support

- Some things are just so important that they should become primitives:

Linguistic Support

- Some things are just so important that they should become primitives:
  - Functions.

Linguistic Support

- Some things are just so important that they should become primitives:
    - Functions.
    - Objects.

Linguistic Support

- Some things are just so important that they should become primitives:
  - Functions.
  - Objects.
  - ...

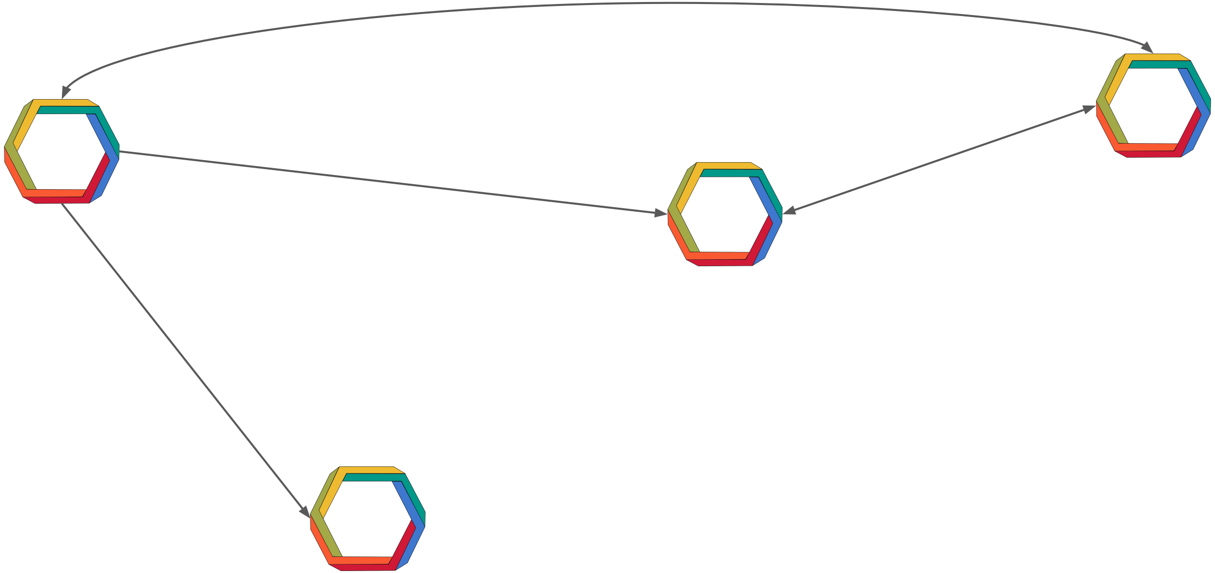
Linguistic Support

- Some things are just so important that they should become primitives:
    - Functions.
    - Objects.
    - ...
    - **Services!**

3 Key Concerns

1. Distribution
2. Integration
3. Technical Debt

Distribution

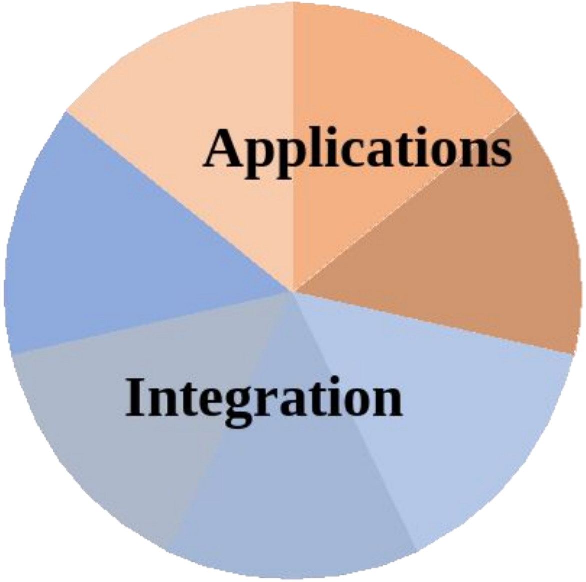


Integration >> applications

Integration costs approximately ⅓ more than applications.

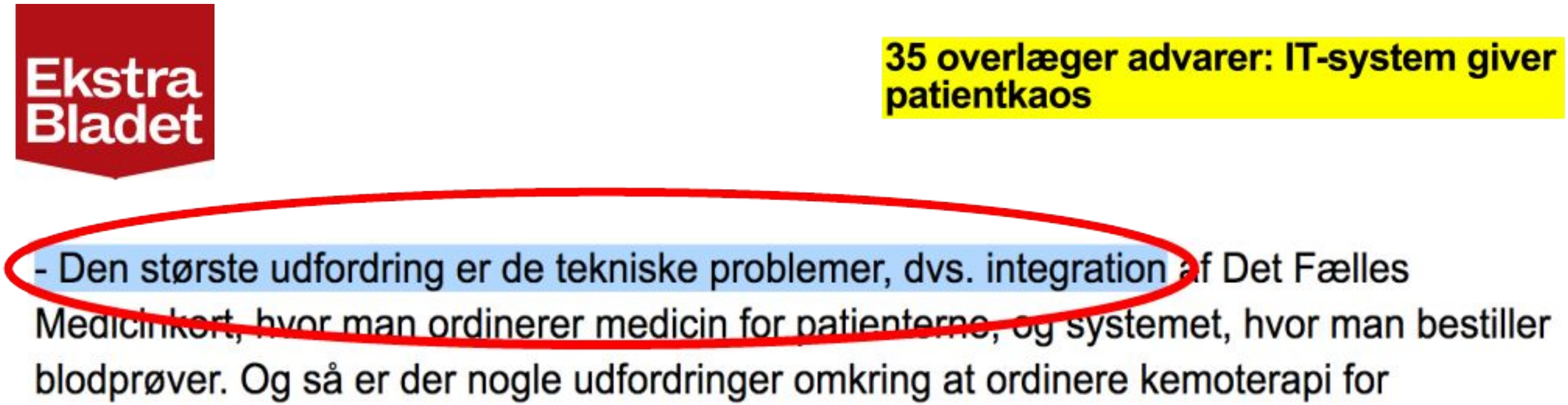
[Forrester Research]

[Gartner]



IT system for health in region Sjælland (Denmark)

- ~150 million EUR invested in 2016, then re-estimated ~375 million EUR. The problem? ***Integration!***



Ekstra Bladet

35 overlæger advarer: IT-system giver patientkaos

- Den største udfordring er de tekniske problemer, dvs. integration af Det Fælles Medicinkort, hvor man ordinerer medicin for patienterne, og systemet, hvor man bestiller blodprøver. Og så er der nogle udfordringer omkring at ordinere kemoterapi for

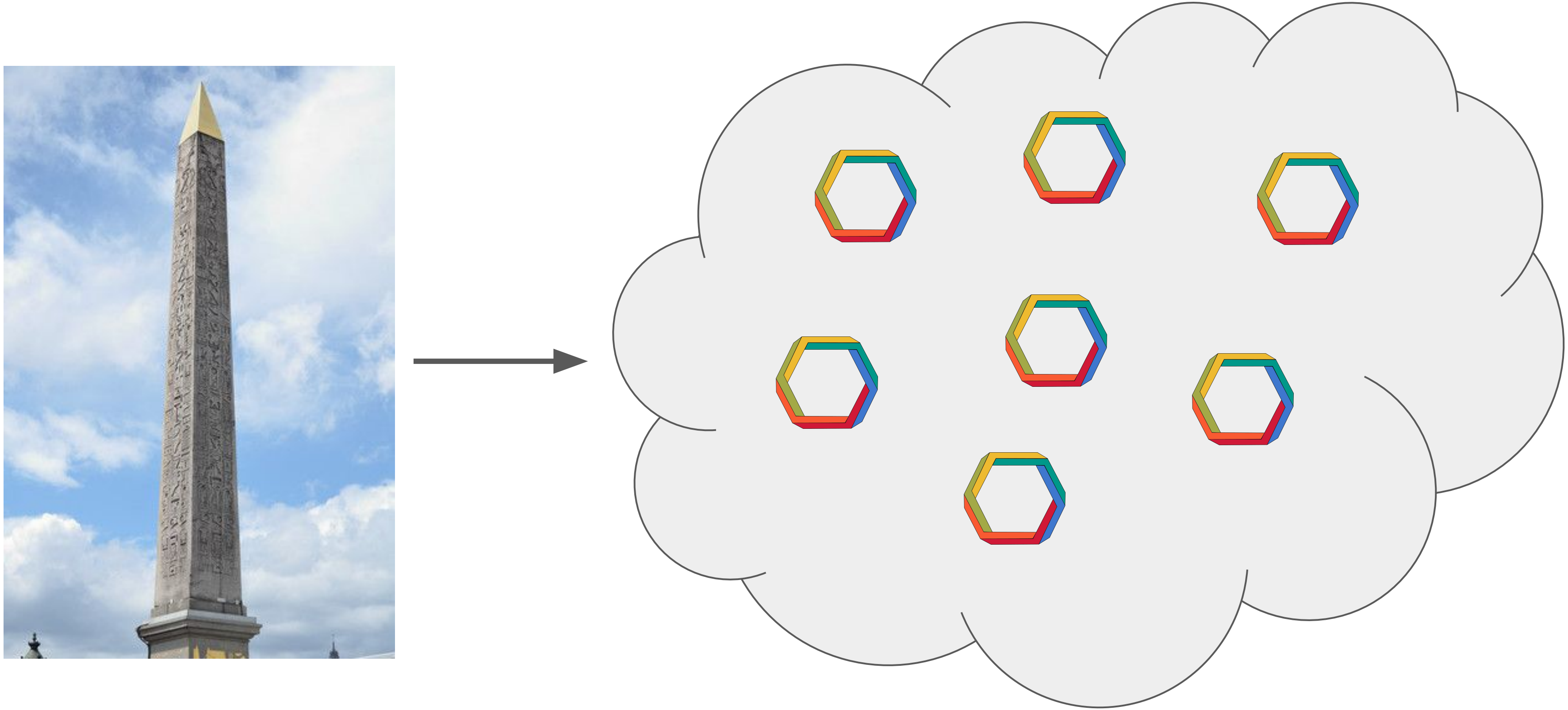
Technical debt

Technical debt

Technical debt

The cost of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer.

Technical debt



Services: what should be native?

- Data Models: How is the data communicated by the service structured?

Services: what should be native?

- Data Models: How is the data communicated by the service structured?
- APIs: What does a service offer?

Services: what should be native?

- Data Models: How is the data communicated by the service structured?
- APIs: What does a service offer?
- Access Points: Where and how can APIs be reached?

Services: what should be native?

- Data Models: How is the data communicated by the service structured?
- APIs: What does a service offer?
- Access Points: Where and how can APIs be reached?
- Communication Behaviours: What communications does a service enact?

Theory

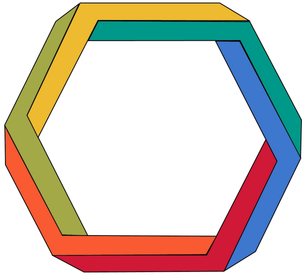
Practice

Theory

Practice

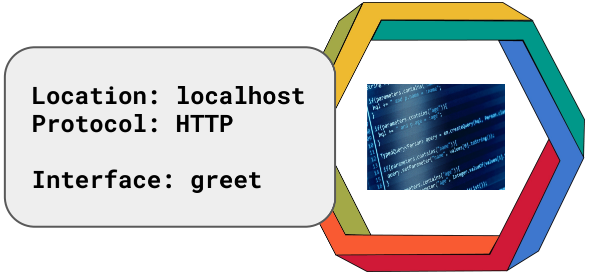
Models of Service Architectures

- Ingredients of services:



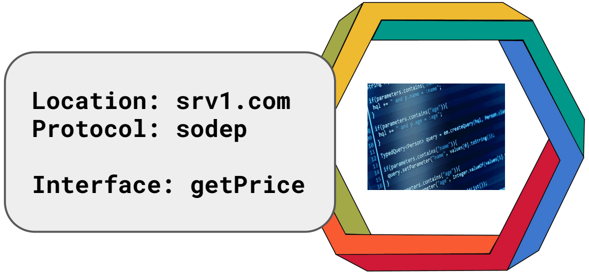
Models of Service Architectures

- Ingredients of services:
  - Data Models.
  - APIs.
  - Access Points.
  - Behaviours.



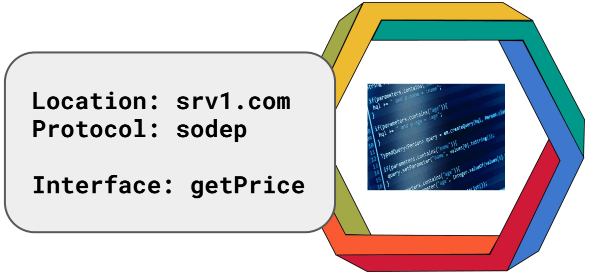
```
Location: localhost
Protocol: HTTP

Interface: greet
```

A Store



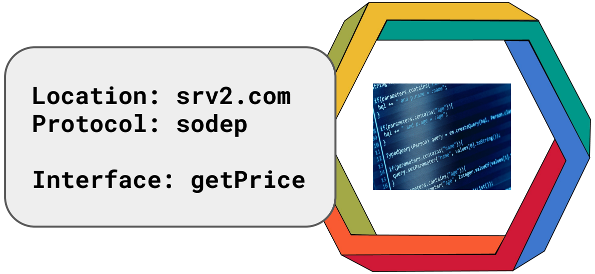
```
Location: srv1.com
Protocol: sodep

Interface: getPrice
```

Two Stores



Location: srv1.com
Protocol: sodep

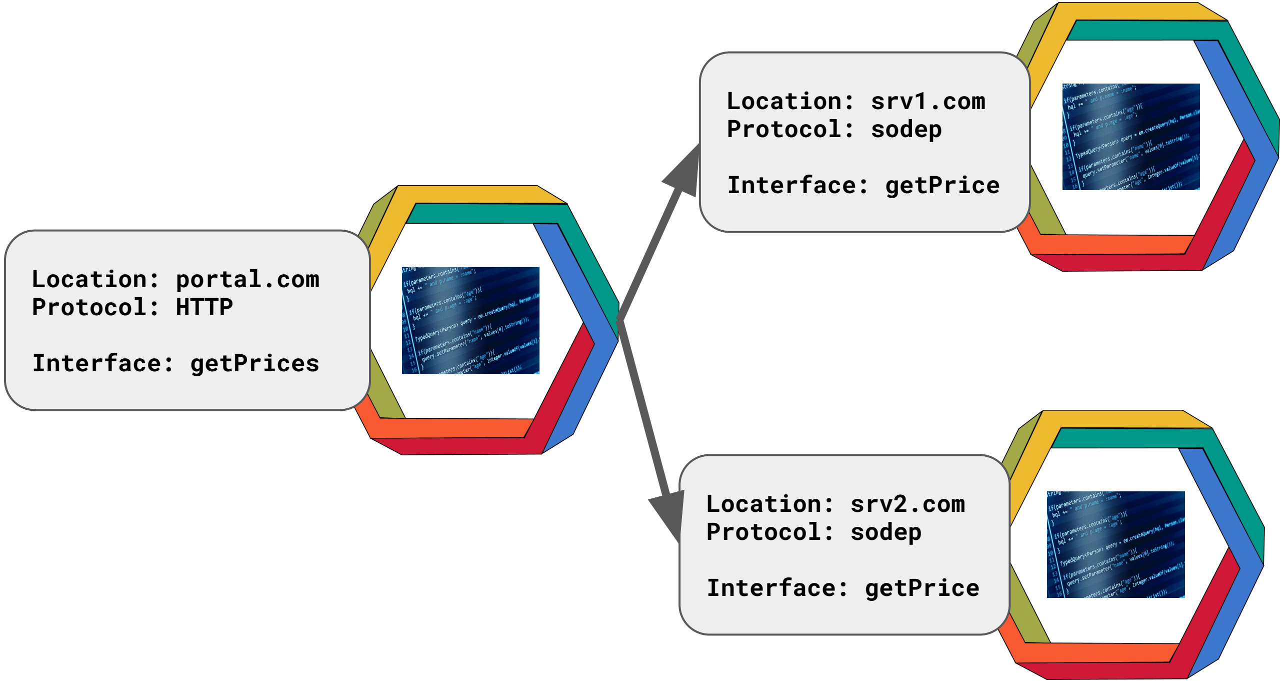
Interface: getPrice



Location: srv2.com
Protocol: sodep

Interface: getPrice

A Portal



Location: portal.com
Protocol: HTTP

Interface: getPrices

Location: srv1.com
Protocol: sodep

Interface: getPrice

Location: srv2.com
Protocol: sodep

Interface: getPrice

Q&A

https://jolie-lang.org