

FS Tech Talk

The Rise of Micro Services



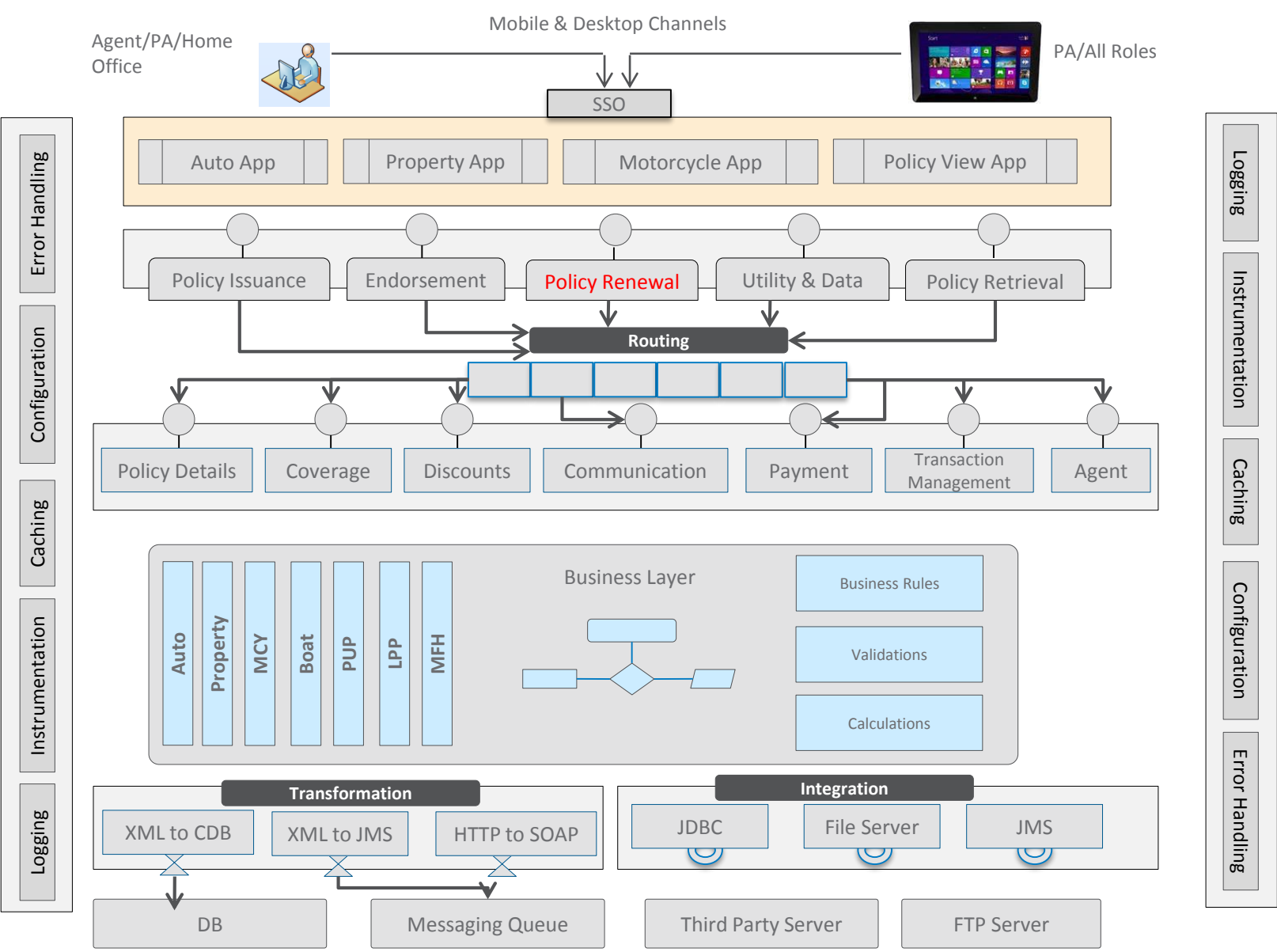
Speaker: Mahesh K Punjabi

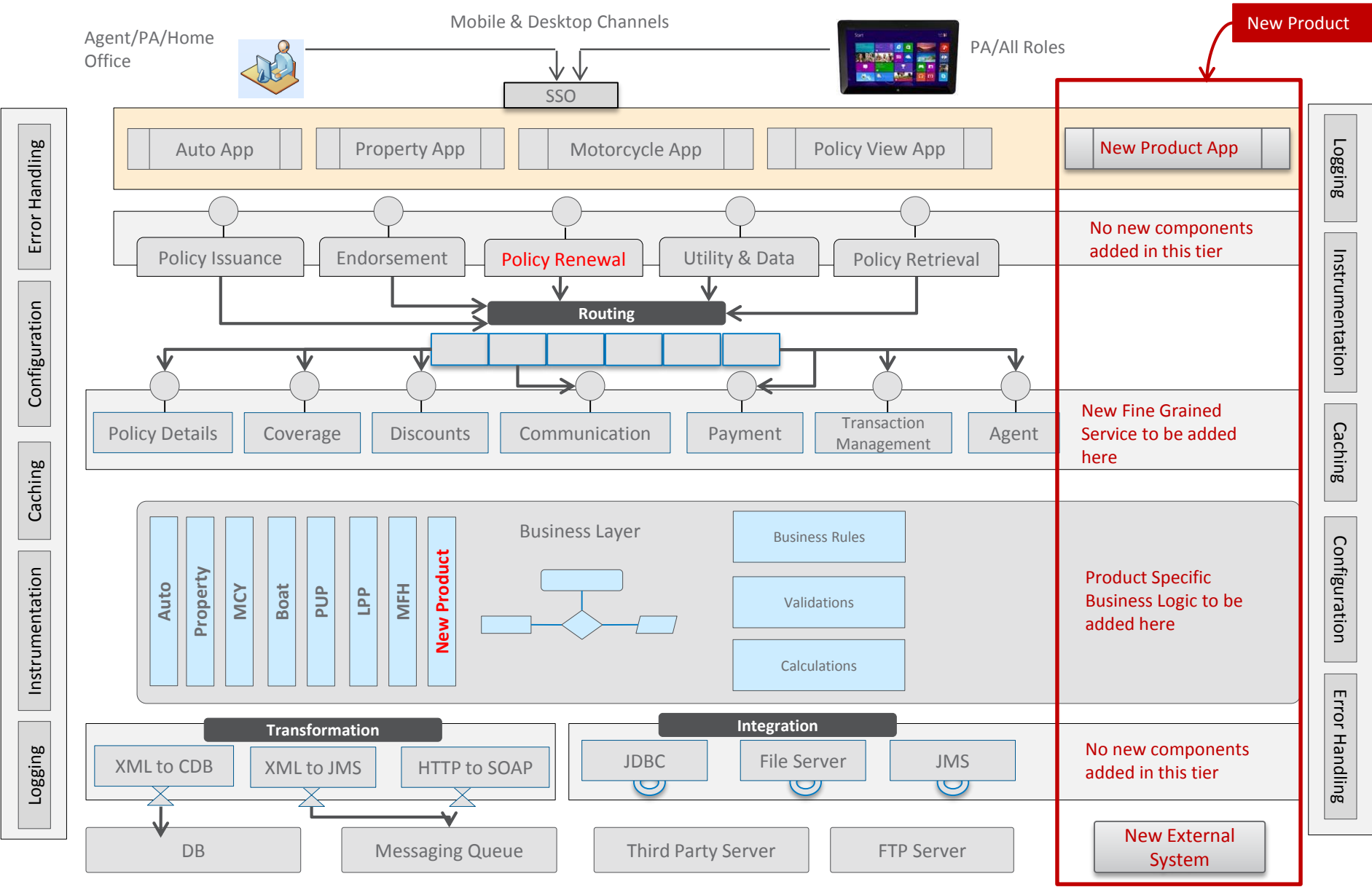
Infosys® | Building
Tomorrow's Enterprise

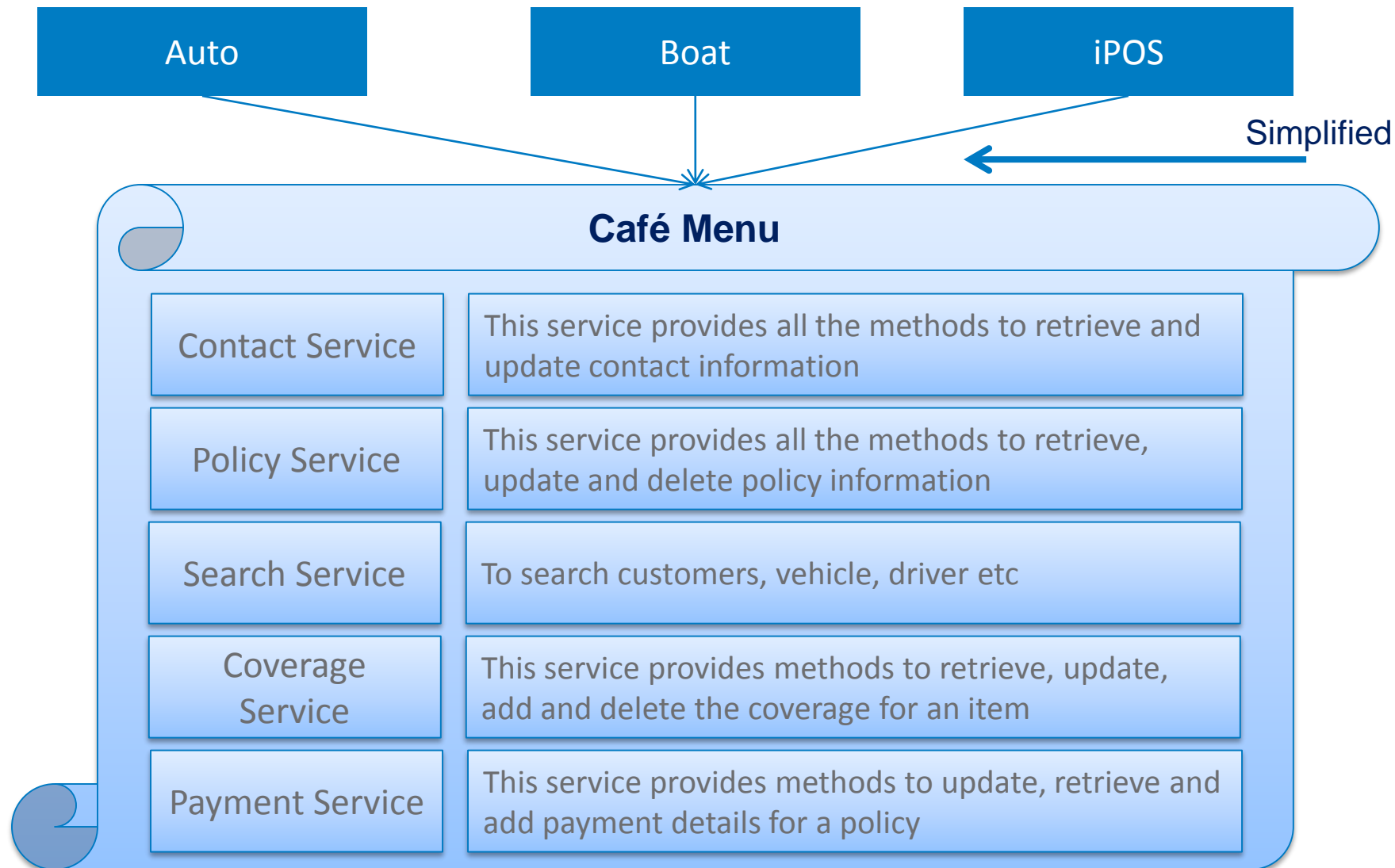
Setting the Context

How I started with Microservices







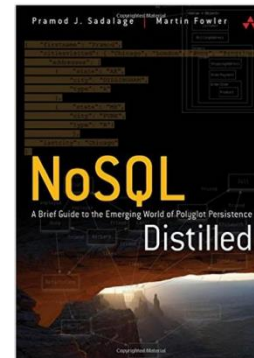
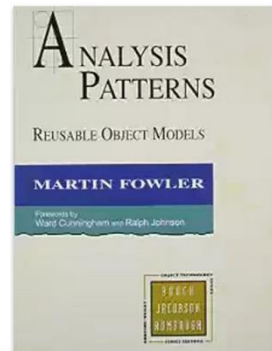
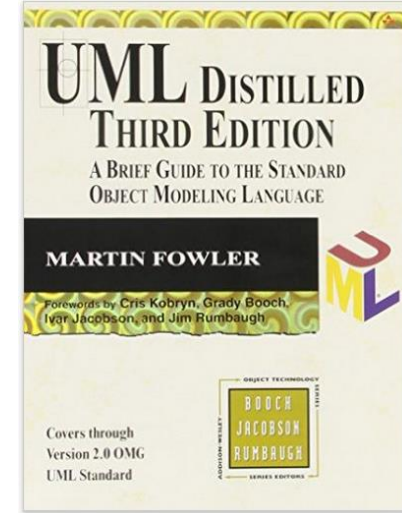
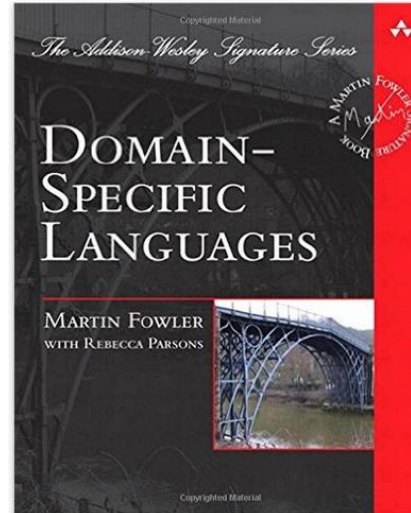
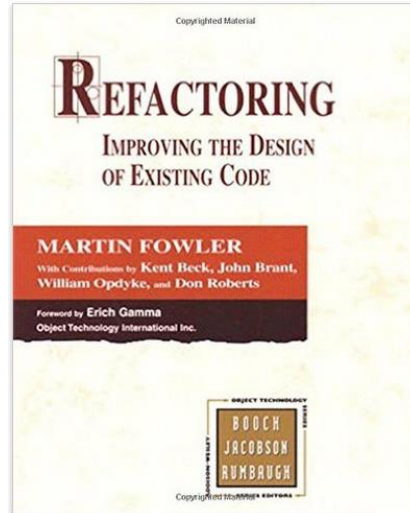
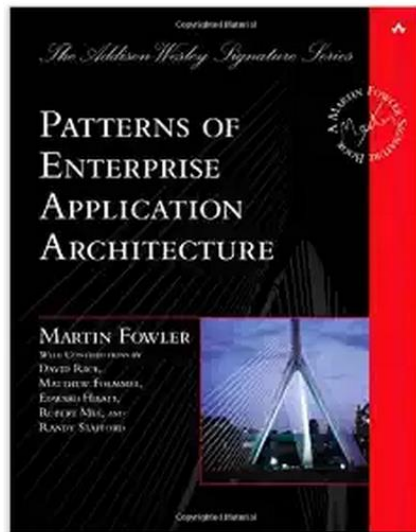




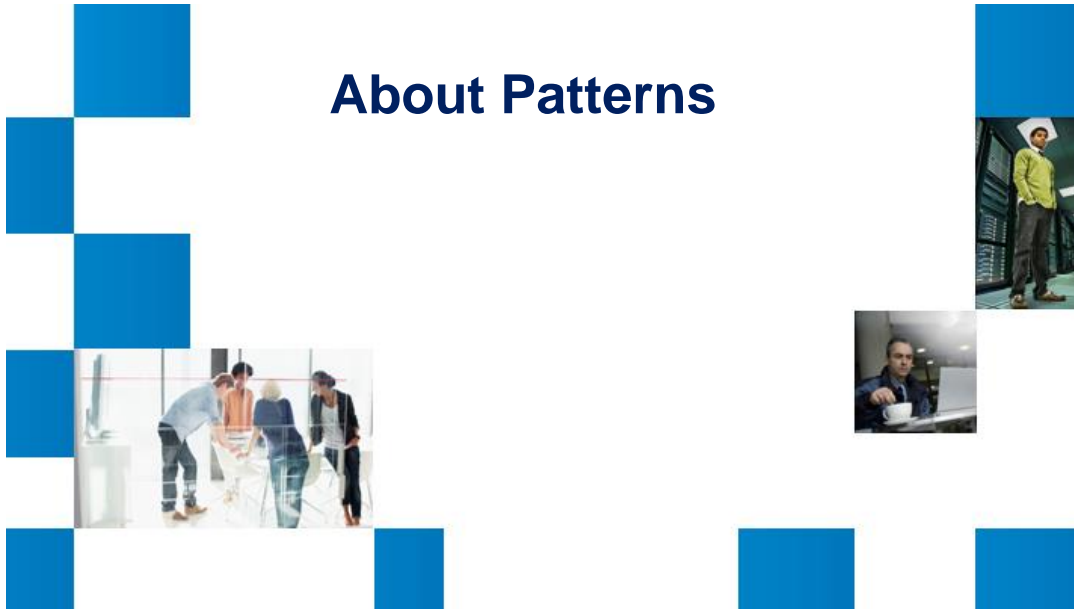
- Author of 7 books on Software Engineering
- Speaker
- Chief Scientist at ThoughtWorks

Martin Fowler

<http://martinfowler.com/>



About Patterns



PATTERN

An idea that has been useful in one practical context and will probably be useful in others

ARCHITECTURE PATTERN

Architectural Style

- Defines structural organization for software systems.
- Specifies the responsibilities of subsystems, rules and guidelines for organizing relationships between them

VS. DESIGN PATTERN?

- Defines how a component must be defined
- Defines how components should be created, how they should communicate, what should be their structure and how should they behave

Architecture Pattern

Pattern Name	Description
Client/Server	Thick Client, Thick Server (most DB)
N-Tier	Presentation, Middle, DB Tiers
Component Based	Decompose application into components
Message Bus	Using messaging for communication between components
Service Oriented Architecture	Design System as services
Layered Architecture	Design system as layers
Hub-Spoke Architecture	Hub will be the center of communication with spokes
ETL	Extraction, Transformation and Loading
Enterprise Service Bus	Have Intelligent Bus to route to multiple endpoints.
Pipe & Filter	Transform information into incremental processes
Reactive Design	Decouples Event from its Processing, Message driven distributed systems
MVC	Model View Controller
Distributed Architecture	Distribute processing logic across several systems

- VS -

Design Pattern

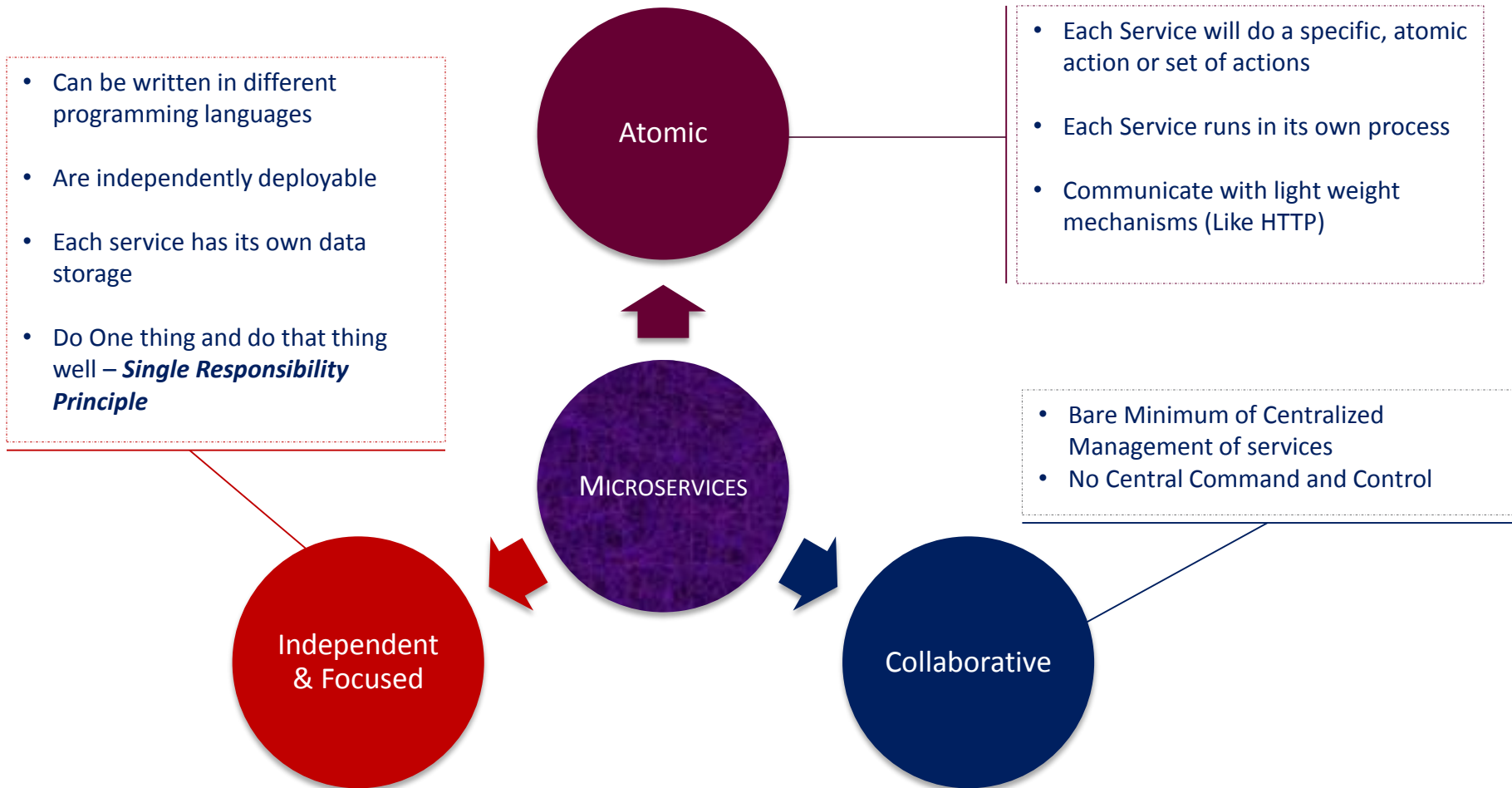
Pattern Name
Creational
Factory Pattern
Singleton Pattern
Prototype
Structural
Adapter
Decorator
Facade
Behavioral
Chain of Responsibility
Iterator
State
Template
Visitor
Observer
Memento

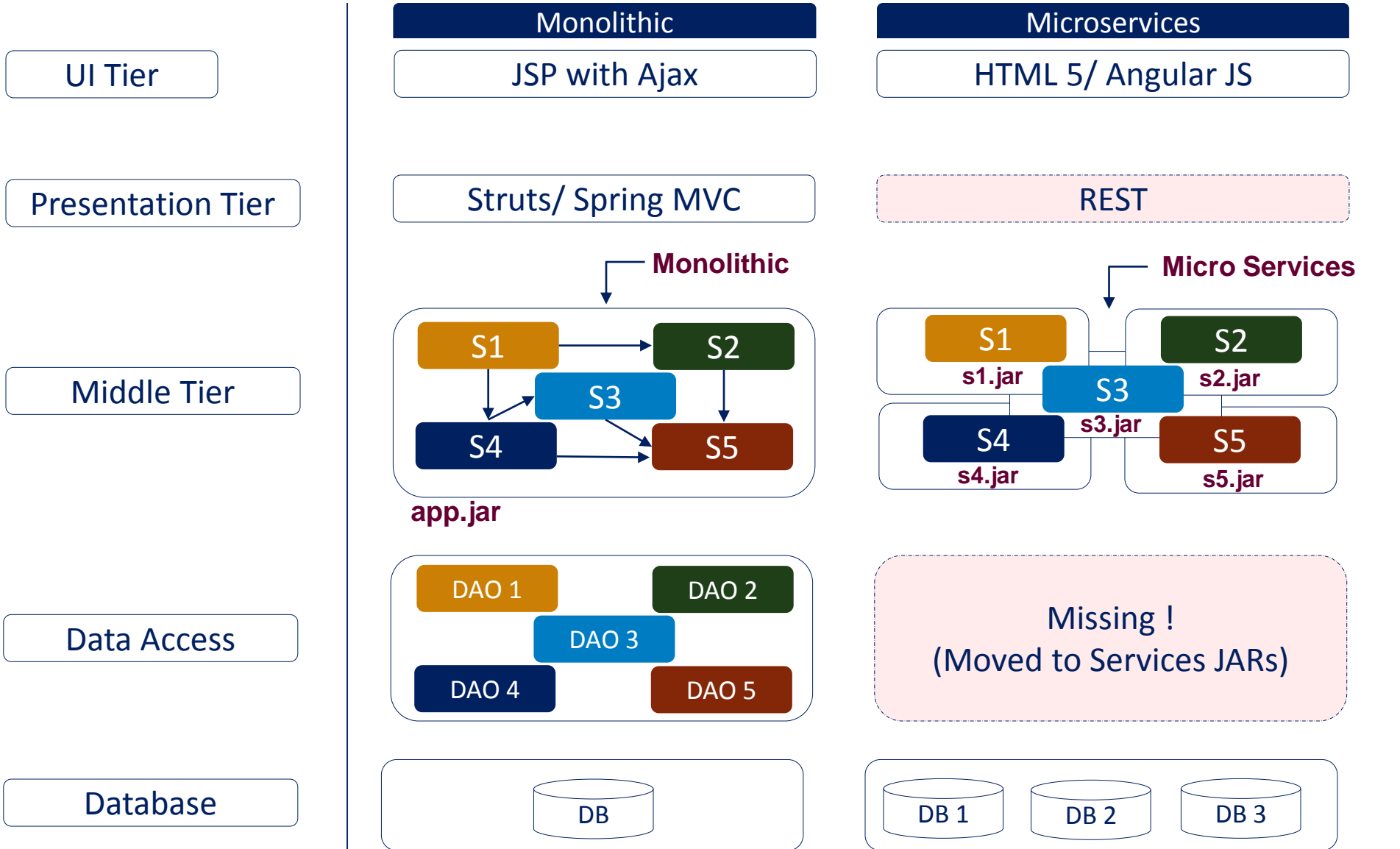
The Rise of Microservices




Microservices is an architectural style of developing an application as a suite of small services, each capable of independently deployable.

- The services are built around business capabilities and independently deployable by fully automated deployment machinery.
- Micro services can even be coded in different languages and will have their own storage





Parameter	Monolithic	Microservices
Packaging	Deployed as an EAR File	Deployed as multiple JAR/Container App 
Deployment	Minor Changes leads to full rebuild and redeployment	Only the service which changed is built & redeployed
Scalability	Scale up monolith by replicating on multiple servers	Scale up only the service which needs to be scaled up
Runtime	Runs as a single process	Each Service runs in its process
Testing	Full Regression Testing	Test primarily the service which was modified
Development	Single Platform (unless distributed SOAP Services)	Each Service <u>can be</u> written in different language



Martin Fowler

We do not claim that the microservices style is novel or innovative. Its roots go back at least to the design principles of Unix. But we do think that not enough people consider microservices architecture and many software developments will be better off if they used it.

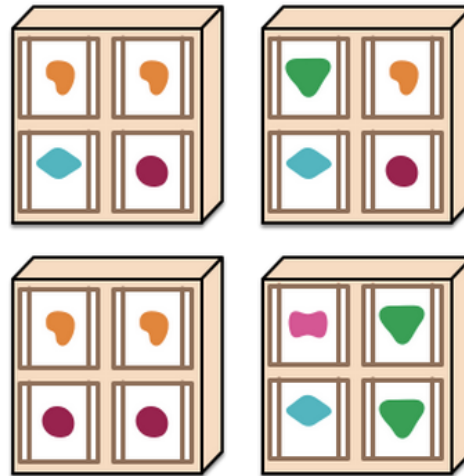
Microservice Characteristics



Component : A unit of software that is independently replaceable and upgradable

Libraries: Components that are linked into a program using in-memory function calls

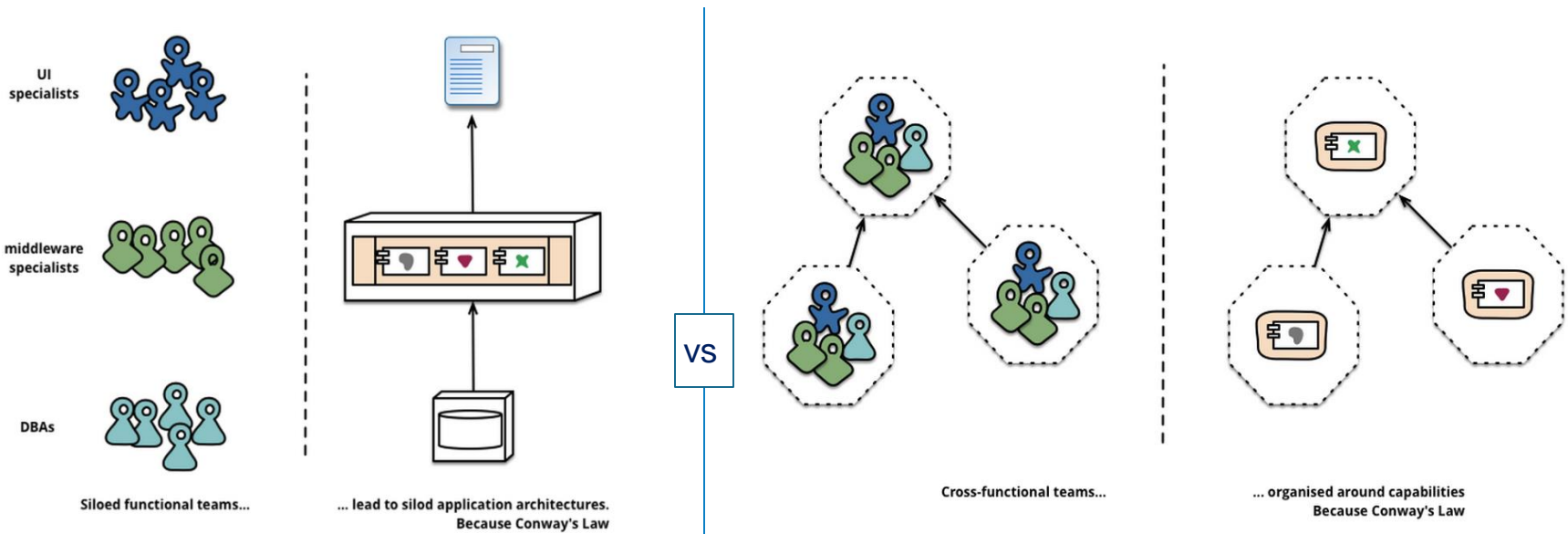
Services: Out of process components who communicate using web service request (like REST) or remote procedure call



By creating services as invocable components, we define clear boundaries for components and minimize dependencies between services

Conway's Law

The structure of the application would reflect the communication structure in an organization **OR**
Any piece of software reflects the organization structure that produced it



How Conway's law created 3 tier architecture systems and we did not know it so far!

How Big is Micro?

- Amazon: Two Pizza Team
 - Typical : 1 Service = 2 folks
- You define your own!**

Microservices approach splits services around business capabilities. Thus service will implement full stack of UI, business logic, persistence storage and external collaboration.

All teams are cross functional, and have full range of skills for development, UX, database and Project Management

3

Products, Not Projects

How Projects are Executed

- Step 1: Assemble Team
- Step 2: Create Software
- Step 3: Test Software
- Step 4: Handover to Maintenance/Prod Support
- Step 5: Disband Team

How Products are Executed

- Step 1: Assemble Team
- Step 2: Create Software
- Step 3: Test Software
- Step 4: Support in Production
- Step 5: Develop bond with users

The Point is

- Product mentality ties in with linkage to business capabilities
- Dev team takes full responsibility of software in production
- Smaller services makes it easy to adopt this approach

4

Smart Endpoints & Dump Pipes

Intelligent Pipes Example : Enterprise Service Bus

- The communication mechanism is very smart
- ESBs have sophisticated facilities for message routing
- ESBs do orchestration and choreography
- ESBs also help do transformation and apply business rules

Dump Pipes in Microservices

- Microservices are as decoupled as possible
- Each Micro service will apply its effect (or filter)
- The orchestration is through simple REST Protocols, rather than
 - BPEL
 - Central Orchestration

The Point is

- Do not use complex orchestration mechanisms like Business Process Execution or ESB Orchestration.
- All the User to drive services through REST calls
- Use REST, Use API. Microservices uses the principles and protocols that world wide web is built on
- Or use light weight messaging like RabbitMQ

5

Decentralized Governance

- Netflix & Amazon have adopted this
- Being woken up at 3am every night by your pager is certainly a powerful incentive to focus on quality when writing your code
- Teams are responsible for all aspects of the software they build including operating the software 24/7
- Use Node.js when you need it, use Angular when you need it. There is no central one solution fits all...

6

Decentralized Data Management

- Each Service manages its own database, either different instances or even different database types
- Use Global Transactions for consistency
- Challenge with performance

7

Continuous Delivery

- Continuous Integration has been superseded by Continuous Delivery.
- Deployment of each service is automated along with automated tests
- Deployment should be a non-event, it should be continuous which is made possible by Microservices

8

Designed for Failure

- Each service considers failure and what are the possible options to counter it
- Microservices are more resilient to failure than the monoliths (Ex: CITCO Experience)
- Microservices lay emphasis on real time monitoring of services
- In monolith its difficult to know if one service is suddenly disconnected
- **Takeaway:** Avoid synchronous calls between services. They are recipe for failure.

Microservices

Overcoming Challenges



1

Performance

Challenge

- Putting one service per deployment and invoking it over the network will lead to huge performance overhead

Solution

- Let the UI tier be the orchestration tier
- Use Single Page/ Single Responsibility principle to compose applications

2

Maintenance

Challenge

- Managing several hundreds deployed versions of services will lead to maintenance nightmare

Solution

- Remove the central command and control mind set
- Let each service be maintained by its own team.

3

Cost

Challenge

- Converting the full application to Microservices will incur significant cost

Solution

- Let the core remain monolithic. Use Microservices for additional features
- Use Microservices for newer modules

4

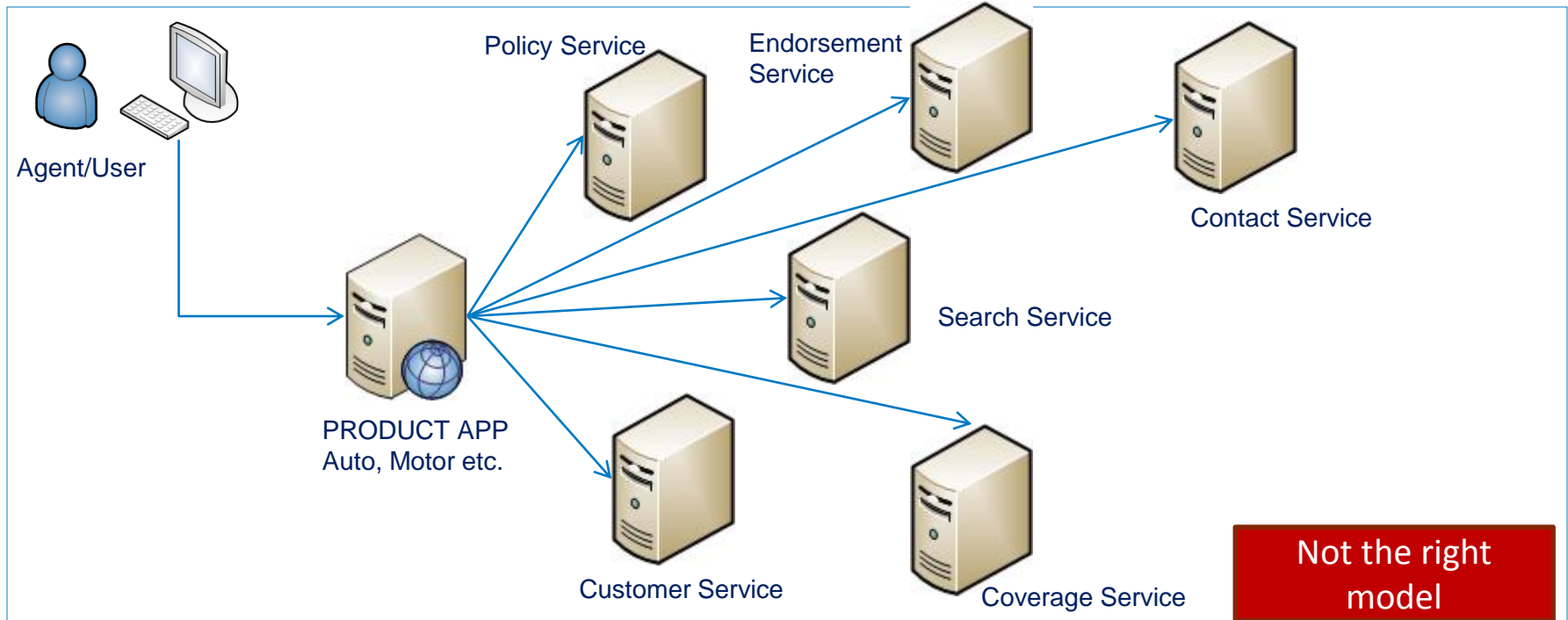
Infrastructure Challenges

Challenge

- The business/technology/operations will not be willing to move to Docker immediately

Solution

- Let the Microservices be created as architecture pattern.
- Let them live in a monolith.
- In future they can be very easily migrated



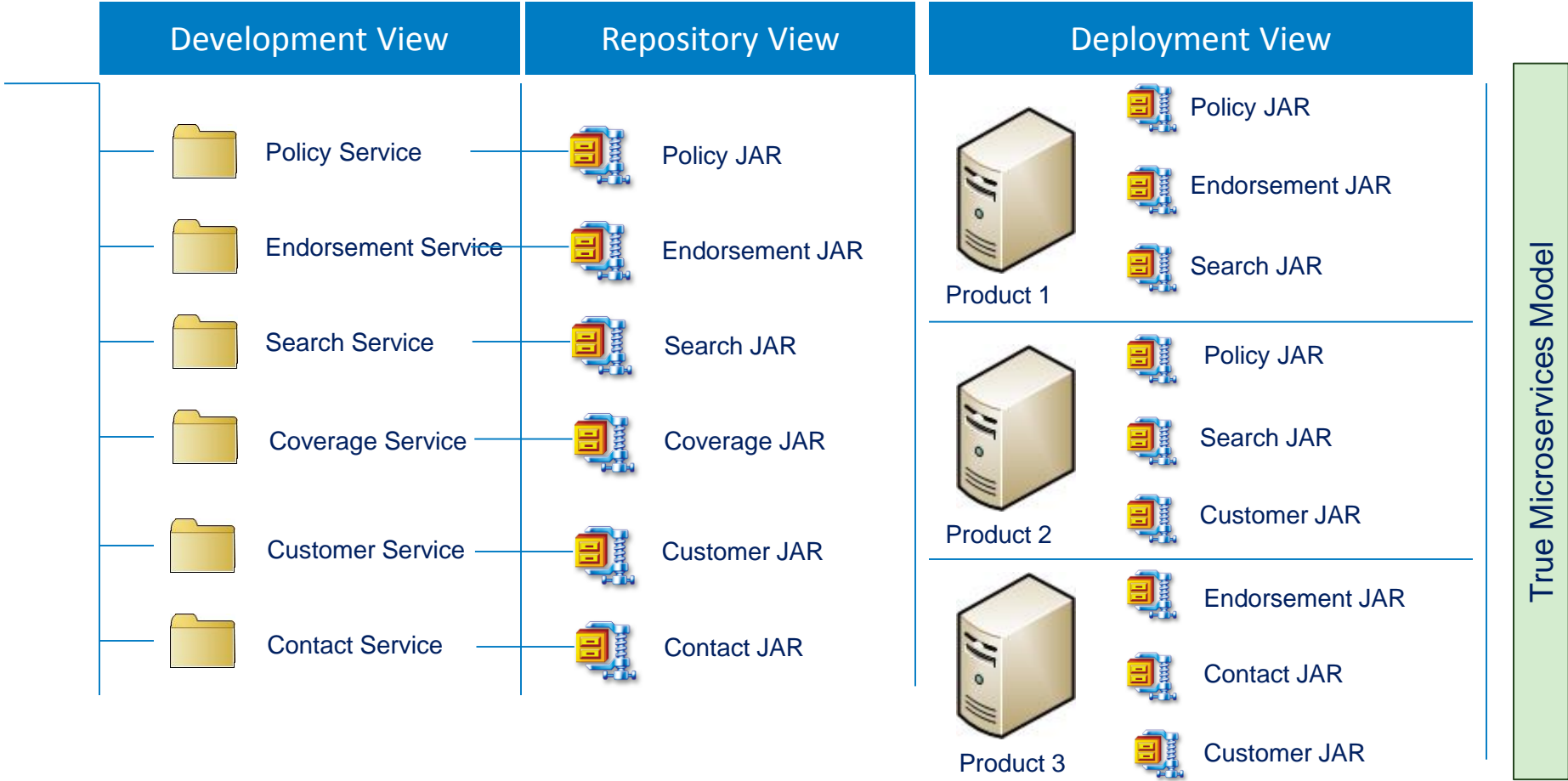
Issues with traditional SOA Model

- Multiple Points of Failure
- Distributed Calls impact performance
- Upgrading a service impacts all the products which need a service
- Difficult to identify and upgrade common components
- Highly Complex, Difficult to maintain and comprehend many times

DEMO

Basic Premise

Separate Service Development from Service Deployment



Microservices

In Financial Services (FS) Unit



1

Amex

Account

- Amex GCS is a Global communication service sending service communication for Amex

What are we doing

- The next generation architecture of GCS will use Microservices.
- Client has shown good interest and has asked his architects to be educated on this. Two sessions were done for client

2

RBS

Account

- Royal Bank of Scotland Bankline Account

What are we doing

- RBS has asked for next generation Bankline Architecture
- Our current architecture is totally modelled around microservices

3

Citi

Account

- Citi, Business Services for Third Party Account Openings

What are we doing

- The client has asked for an exclusive proposal on Microservices.
- We are responding to client's questions

4

Misc

Account

- GS and Others

What are we doing

- Engaging with the clients
- Providing architectures
- Consulting

1

Talk to Clients

- Talk to clients if they would be interested in Microservices.
- Gauge their interest and we can conduct an informative session with the client

2

Present Tech Note

- The tech note on Microservices will be available shortly.
- The note can be shared with the clients

3

Do POCs

- If there is an opportunity to prove to the clients how the Microservices can be fit in their scenario, we can do a POC for them first.

4

DO RFPs

- Engage STAR Architects for RFPs where Microservices can be a good fit

Microservices

Tools, Books & References





Dropwizard is a Java framework for developing ops-friendly, high-performance, RESTful web services.

Dropwizard pulls together **stable, mature** libraries from the Java ecosystem into a **simple, light-weight** package that lets you focus on *getting things done*.

Dropwizard has *out-of-the-box* support for sophisticated **configuration, application metrics, logging, operational tools**, and much more, allowing you and your team to ship a *production-quality* web service in the shortest time possible.

<https://dropwizard.github.io/dropwizard/>

PAAS



API Tools



Containers



Manage a cluster of Linux containers as a single system to accelerate Dev and simplify Ops.

Integration/
Messaging/
Miscellaneous
Frameworks



FREE EBOOK



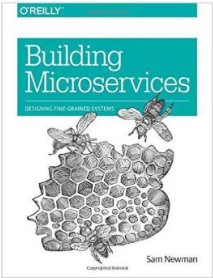
<http://www.oreilly.com/programming/free/software-architecture-patterns.csp>

Case Studies

How Netflix Handled Fault Tolerance with Microservices

<http://techblog.netflix.com/2012/02/fault-tolerance-in-high-volume.html>

Books

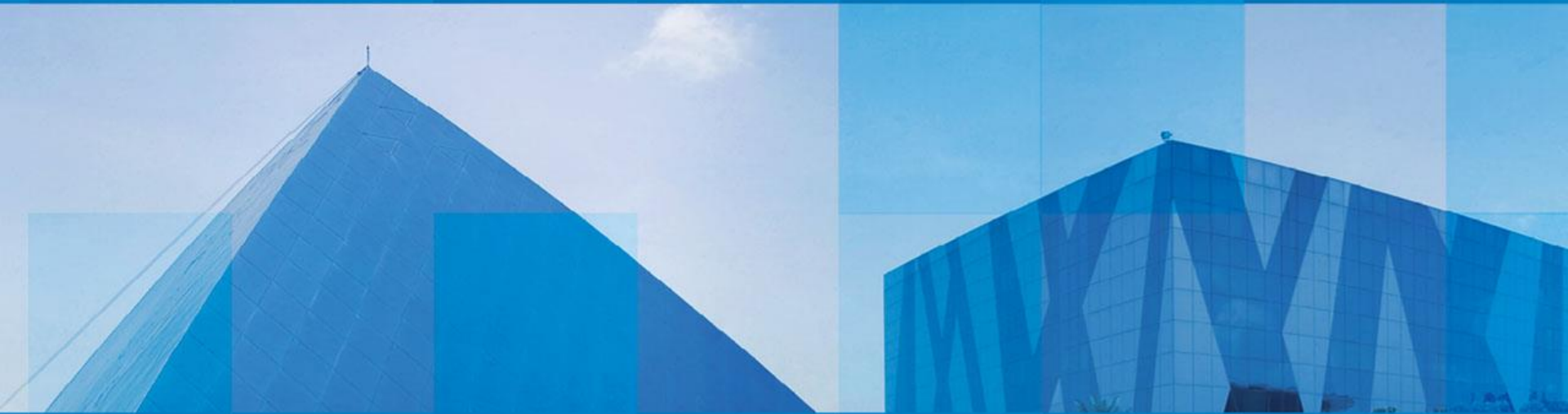


Redbooks



Microservices from Theory to Practice
Creating Applications in IBM Bluemix Using the Microservices Approach

Thank You



© 2013 Infosys Limited, Bangalore, India. All Rights Reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.

Infosys® | Building
Tomorrow's Enterprise