A Gentle Introduction to

MICROSERVICES

From Theory into Practice

LEMİ ORHAN ERGİN
Agile Software Craftsman

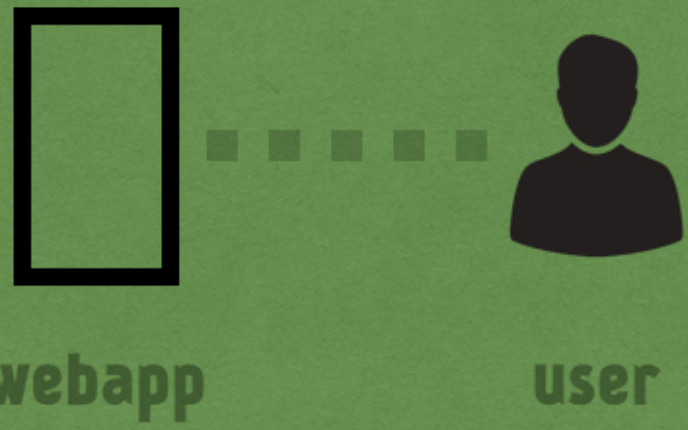
Agile Software Craftsman
Scrum, Kanban ve XP Practitioner
Passionate Developer, Leader
Sony & eBay Alumni

@lemiorhan
lemiorhanergin.com
@lemiorhan

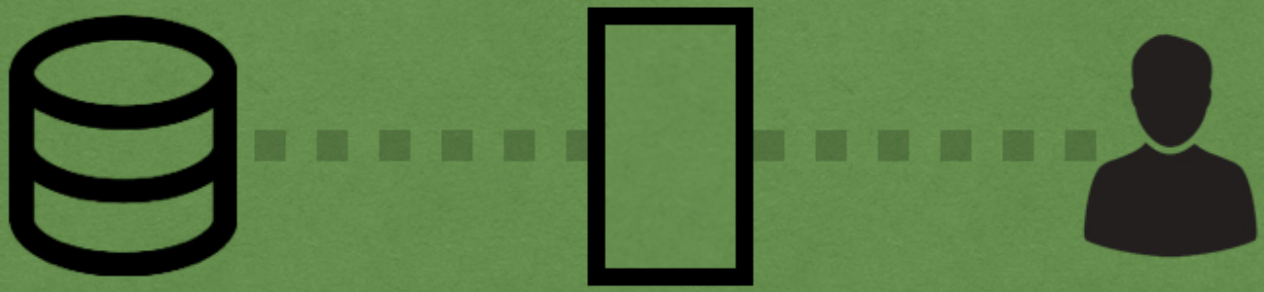
LEMİ ORHAN ERGİN

FROM THE VERY BEGINNING

LET'S BUILD A WEB APP



I NEED DATA



databases
services
file system
data sources

webapp

user

ALL FEATURES IN ONE

databases
services
file system
data sources

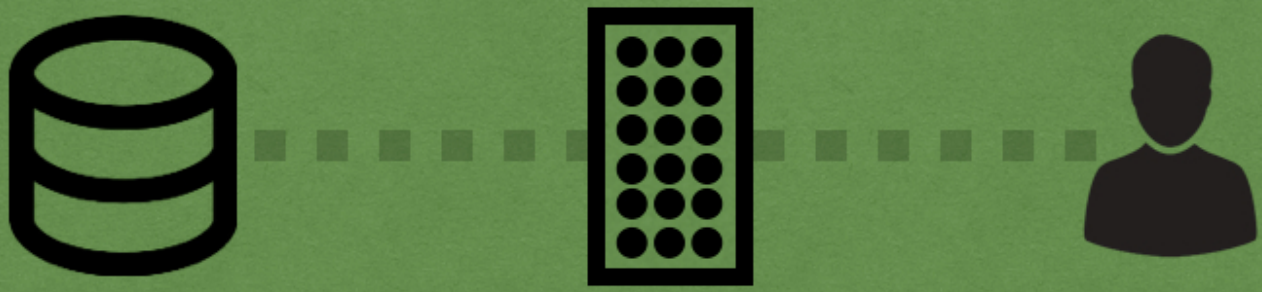
Data access logic
Presentation logic
Shopping chart
Content retrievals
Authentication
Input validation
Shipping information

user

SPAGETTI CODE

Everything is all
together
in a rubbish can

IMPROVE THE INTERNALS

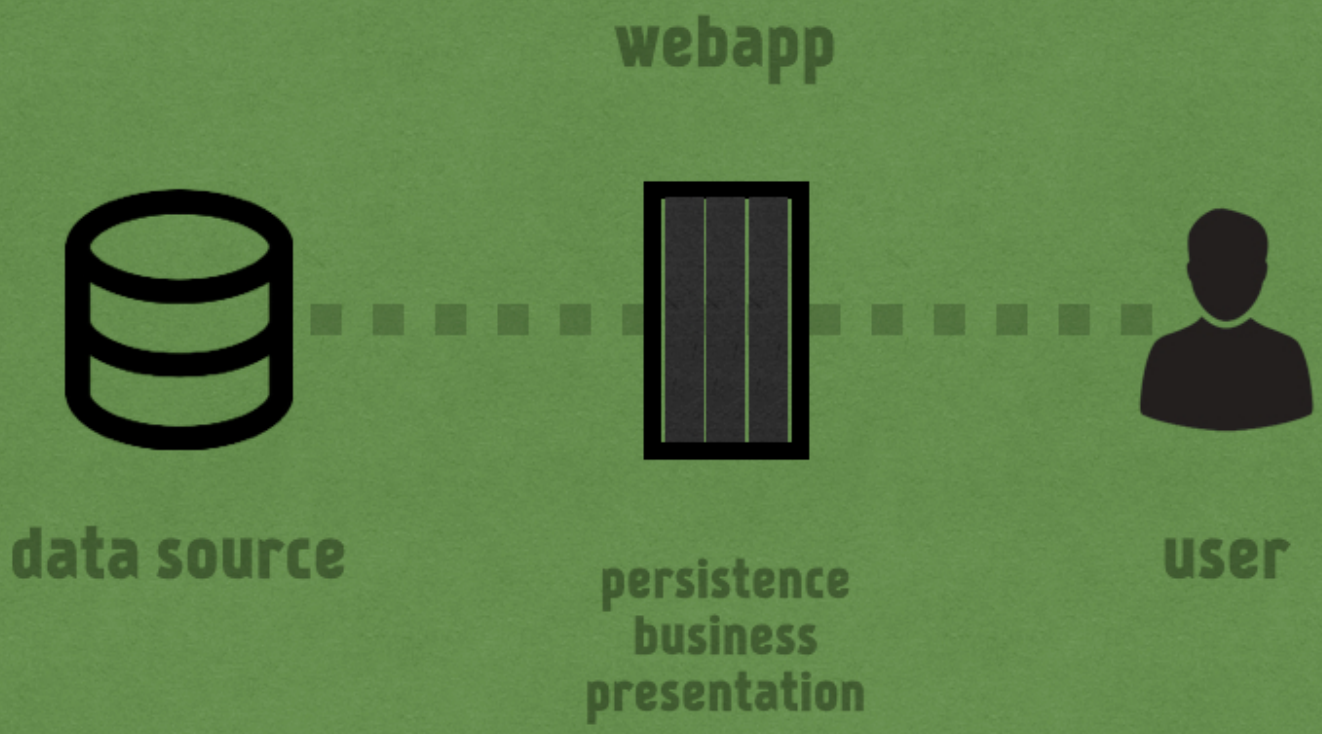


data source

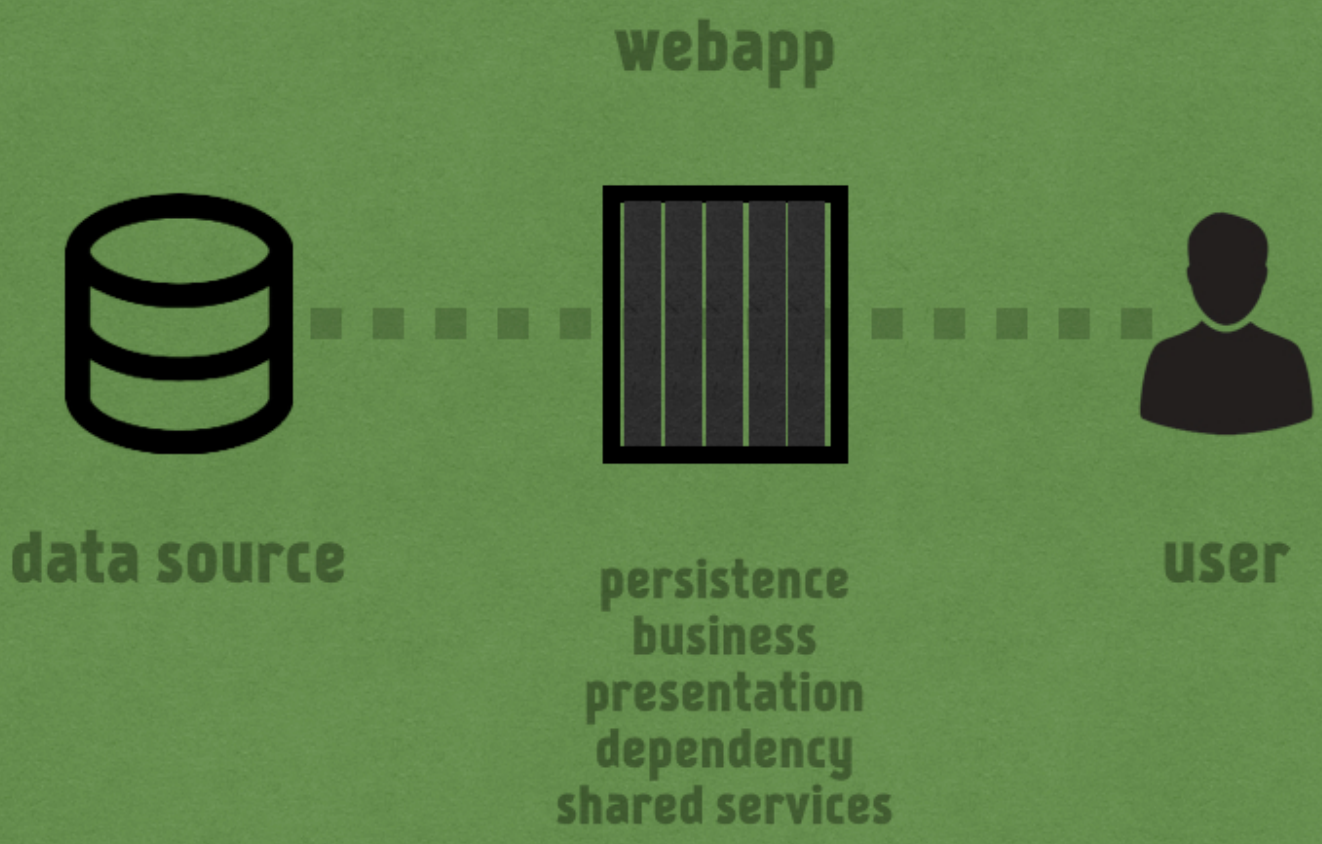
object oriented design
modular design
design patterns

user

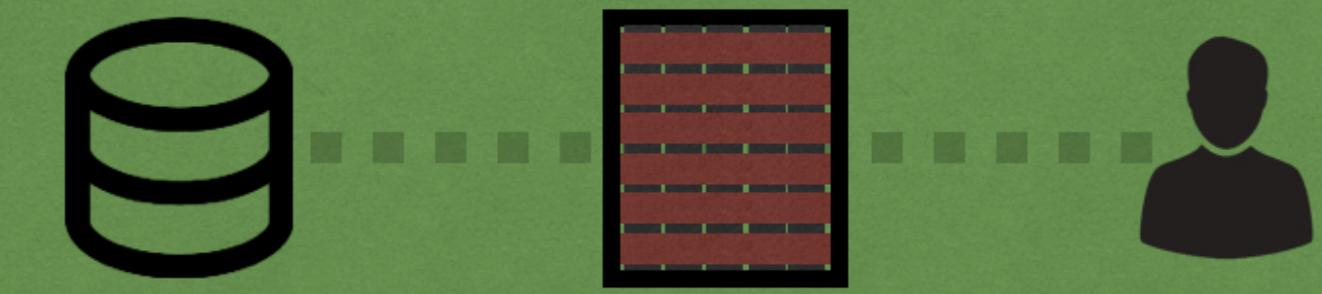
LET'S MAKE IT LAYERED



ADD MORE LAYERS



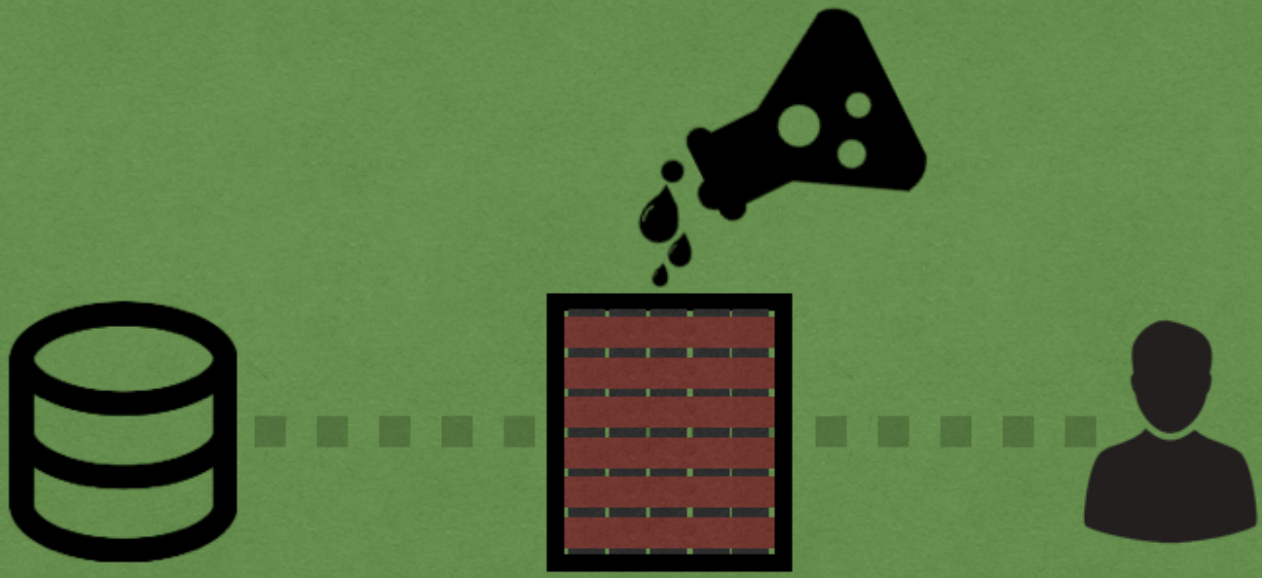
ADD NEW FLOWS & FEATURES



BAKLAVA CODE

Maintainability
Agility / Velocity
Lead time

SCALE VERTICALLY

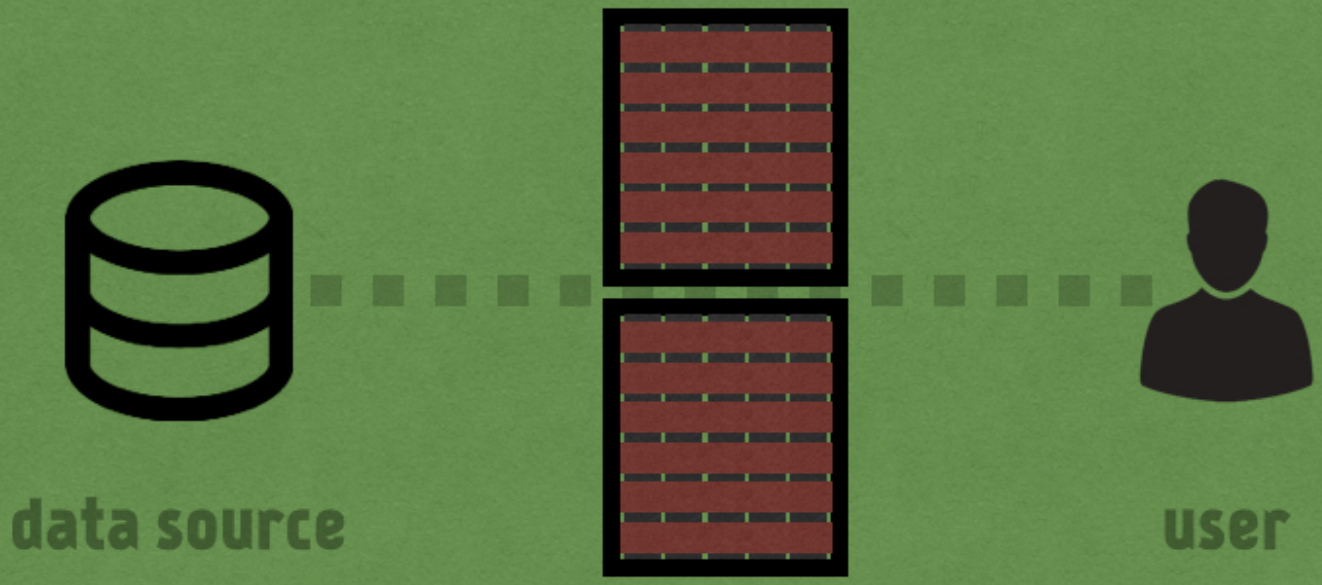


data source

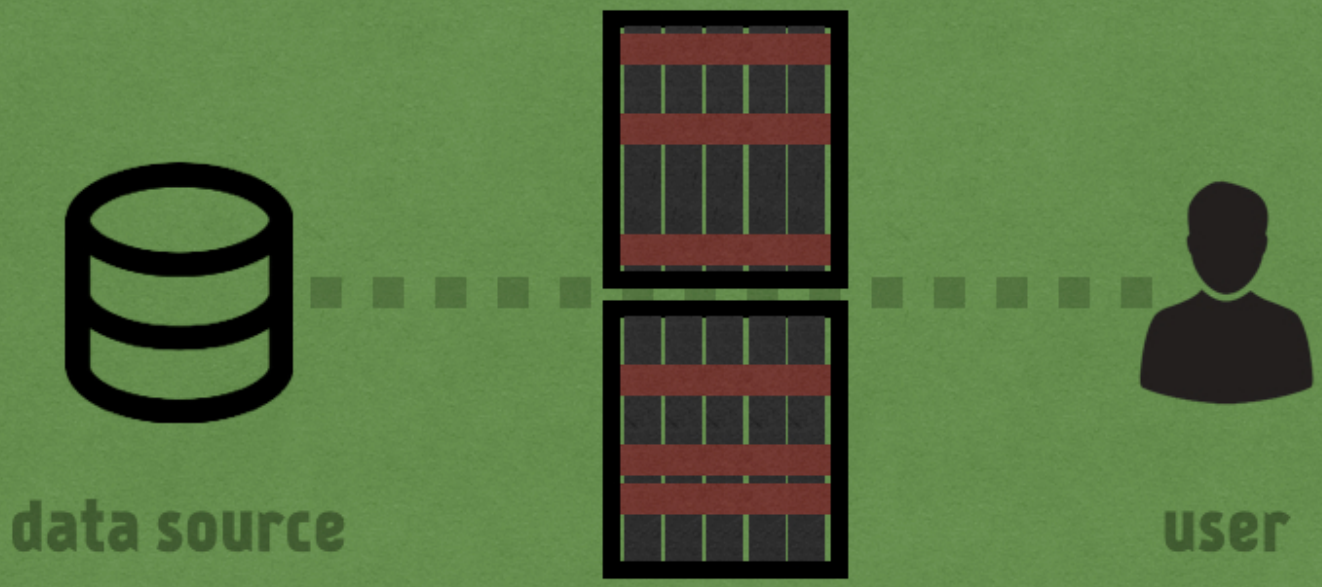
CPU
Memory
Disc Size
GPU
Config

user

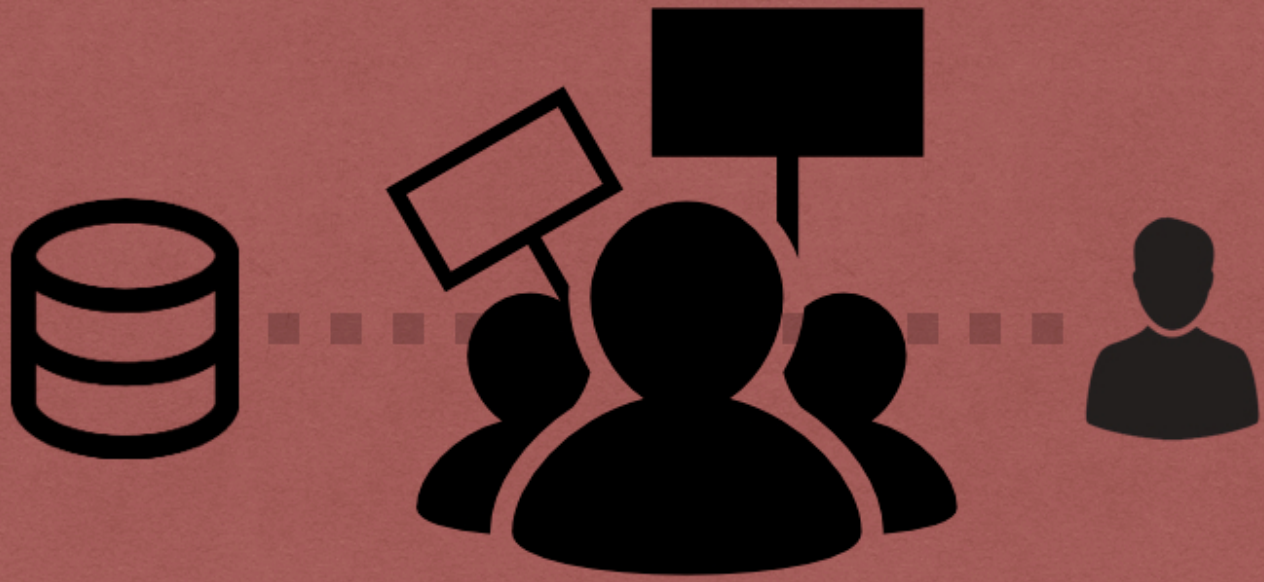
SCALE HORIZONTALLY



SPLIT FEATURES

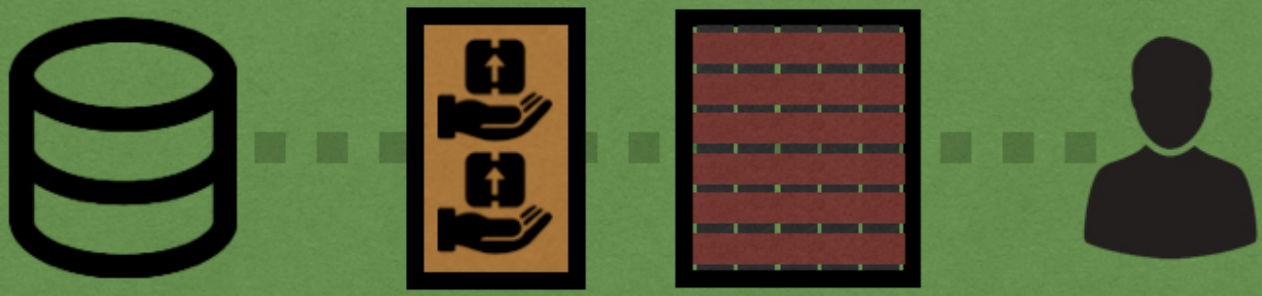


NEWER ISSUES



Load Balancing
Scalability
Deployment
Fault Tolerance
Dependencies
Security

SPLIT TO SERVICES



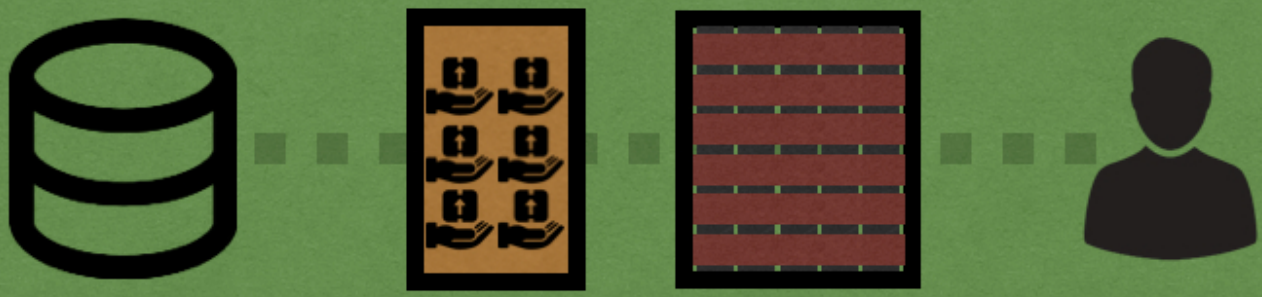
data source

user

SERVICE ORIENTED ARCHITECTURE

Software design and software architecture design pattern based on distinct pieces of software providing application functionality as services to other applications

ADD MORE SERVICES



data source

Even services can be designed monolithic and we can still call them service oriented

user

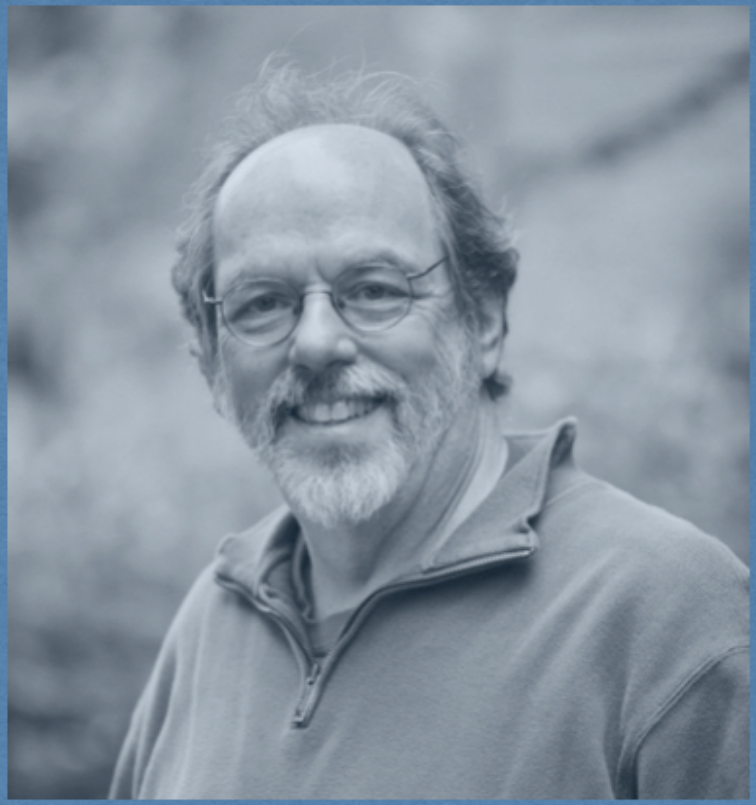
MORE, MORE & MORE ISSUES

Slow development
Irrelevant works
Scalability issues
Big fat services
Endless restarts
Reusability issues
Missing refactoring

WE
BUILD
GODZILLAS

Bad sense of asthetics
Procrastination of refactoring
Premature optimization
Not writing for reuse
Tunnel vision to one vision

THE MONOLITH



MONOLITHIC DESIGN

I don't see how this differs from "badly-written software" or "poorly-factored software" in general.

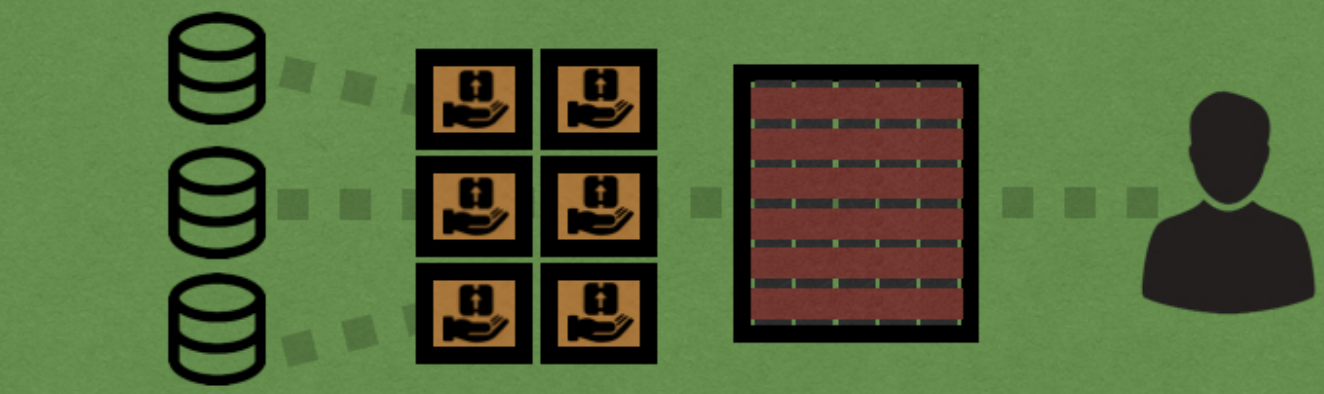
Ward Cunningham

SCALABILITY

Good design scales up well

Michael Ibarra

SELF POPULATED SERVICES



data source

You can define micro services
for your backend functinalities

user

SERVICES EVERYWHERE

MICRO SERVICES

It is a way of designing software applications as suites of independently deployable services

IT'S NOT THE SILVER BULLET

It is not suitable in many situations
Be aware of your needs and think wisely

MICRO SERVICES VS SOA

Micro services is a specific flavour of SOA. Due to unique features, it deserves a name.

Martin Fowler

ARCHITECTURE

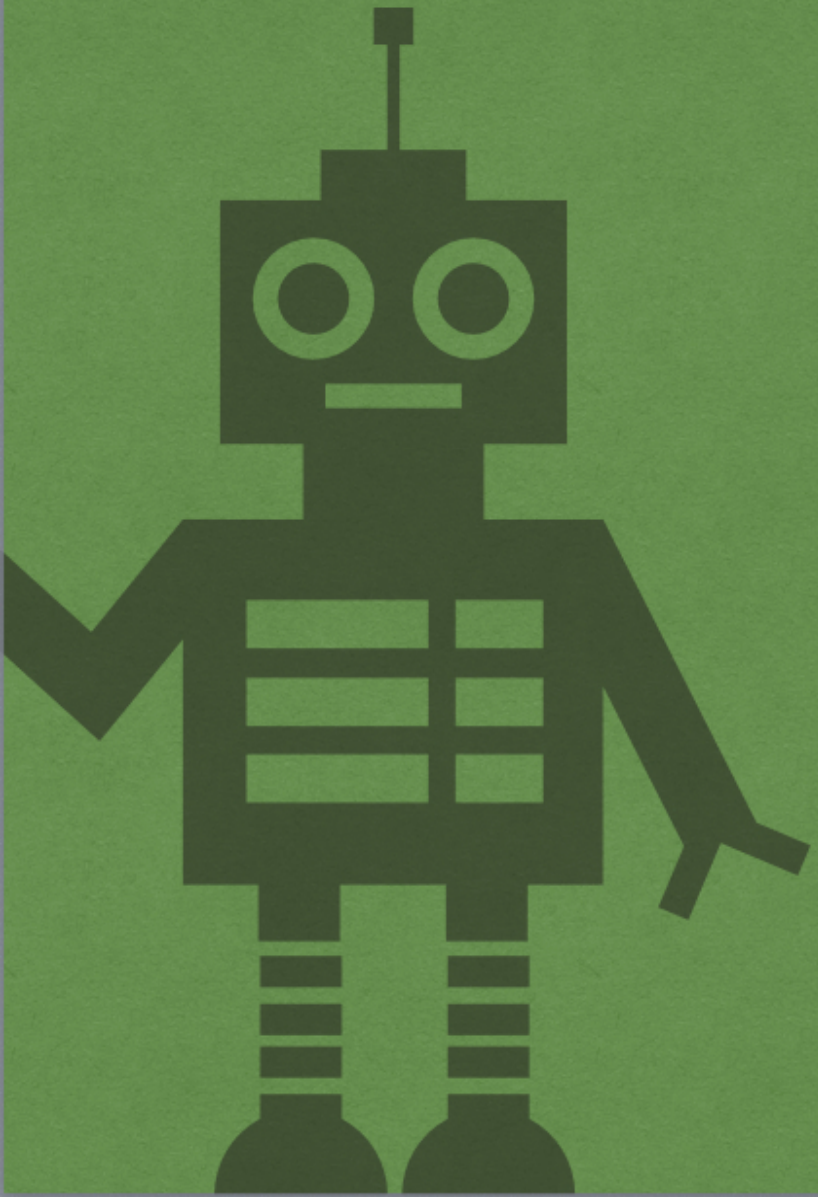
Minimize service dependencies
Evolution mechanism in service contracts
Split features around business capabilities
Keep logic not on layers, on everywhere

COMMON CHARACTERISTICS

Independent

Build
Deployment
Data Management
Business Domain

COMMON CHARACTERISTICS



SOLID Design

Single Responsibility
Open / Closed
Liskov Substitution
Interface Segregation
Dependency Inversion

COMMON CHARACTERISTICS

Wise Decisions

Unix Philosophy
Loosely Coupled
Decentralized
Stateless
Asynchronous
Centralized Log Management

COMMON CHARACTERISTICS

Technology

Can in any language
Rewrite is practical
Use open standards

COMMON CHARACTERISTICS

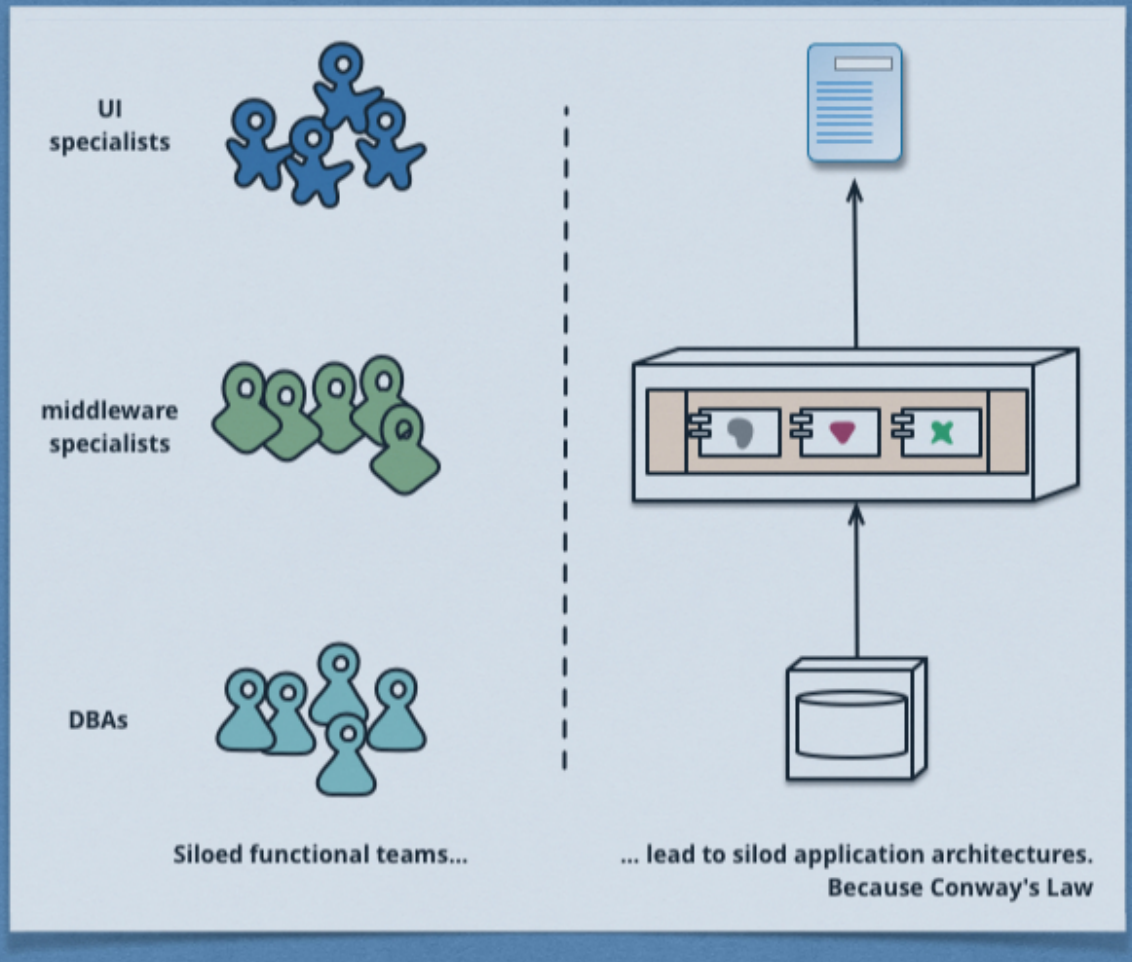
Agile

Fast cycle time
Focused on value
Responsive to change
Emergent design
Short feedback loop
Easier to experiment

CONWAY'S LAW

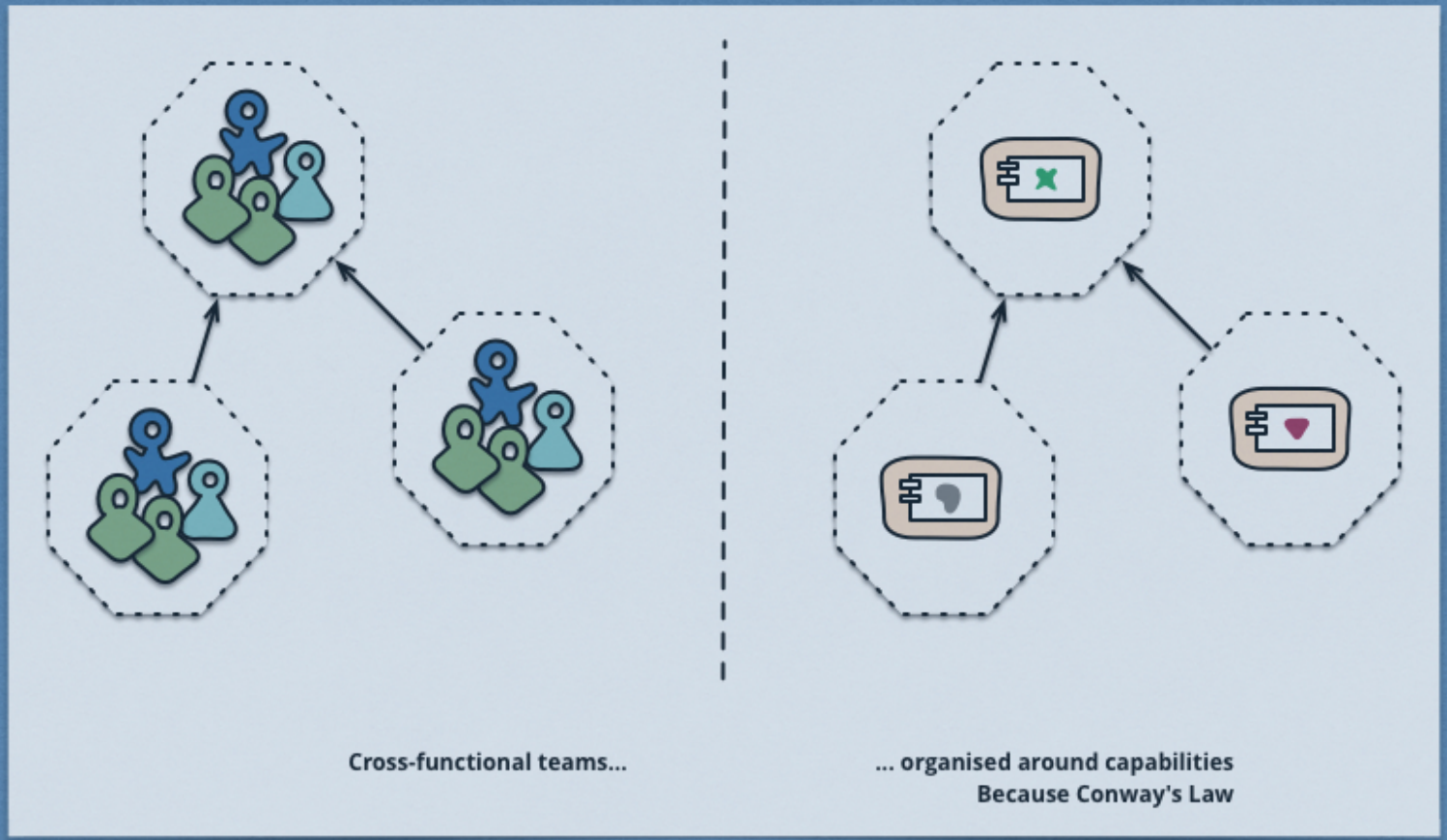
Organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations

Melvin E. Conway



Referenced from Martin Fowler's article

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



Referenced from Martin Fowler's article
http://martinfowler.com/articles/microservices.html

COMMON CHARACTERISTICS

Empowered Teams

Own the product
Have full responsibility
Two pizza team
You build, you run it

WHEN YOU ARE READY?

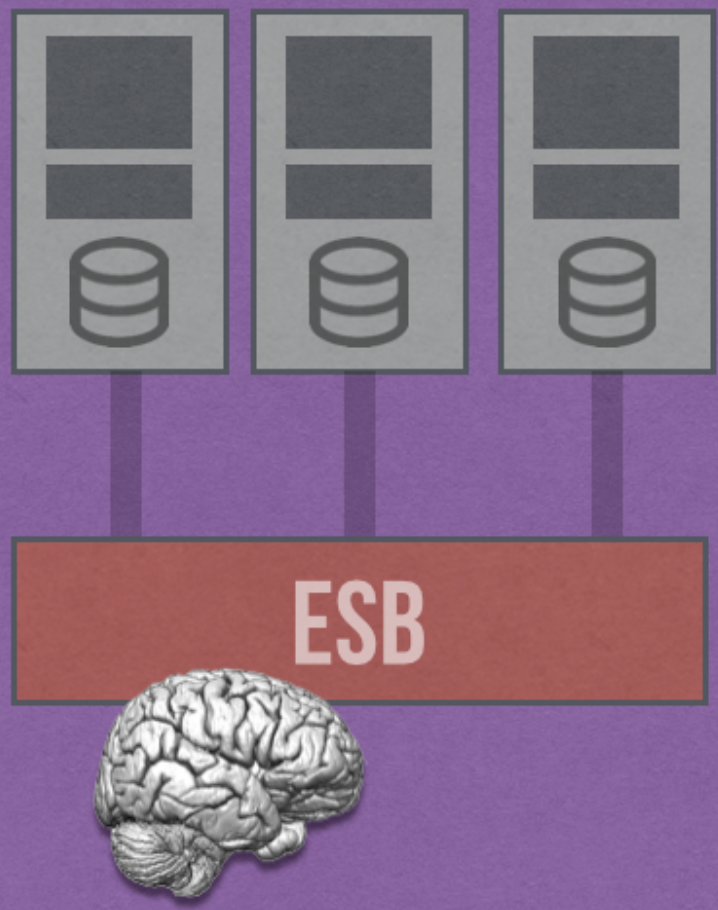
Rapid provisioning
Basic monitoring
Rapid development
Devops culture

HOW BIG

As big as it fits in your brain
Avoid nano-services

HOW TO IMPLEMENT

Avoid using ESBs

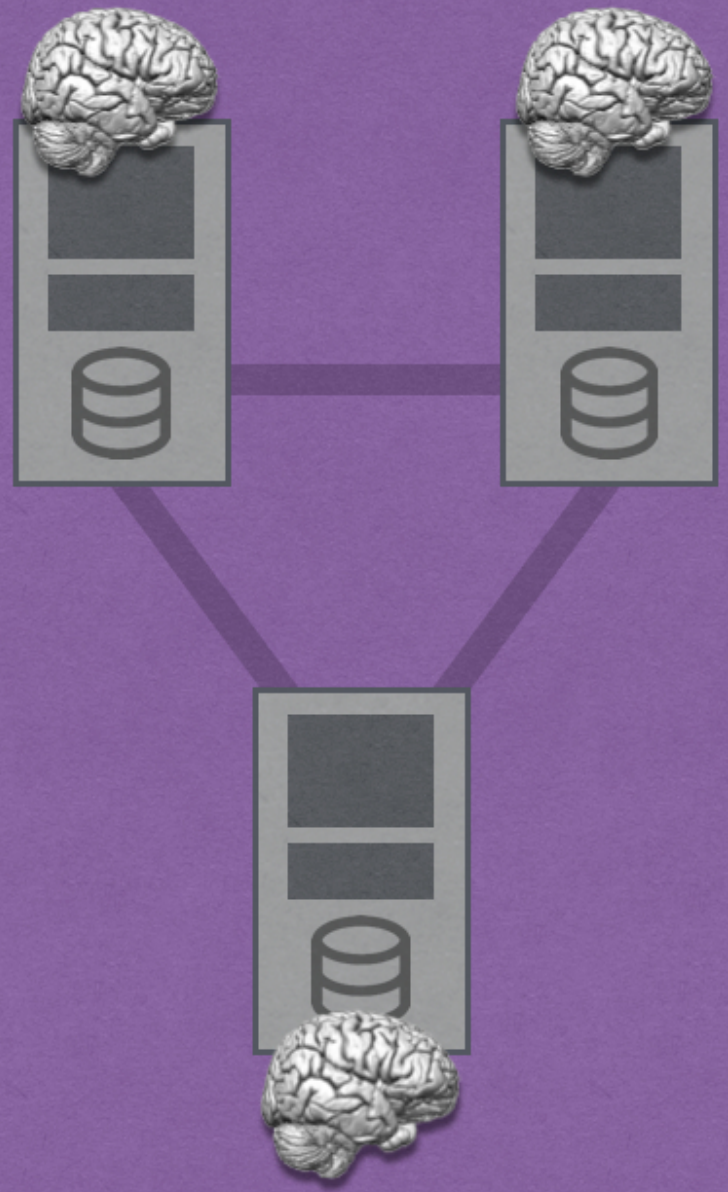


HOW TO IMPLEMENT

Keep services have the logic
Intelligence in the endpoints

HOW TO IMPLEMENT

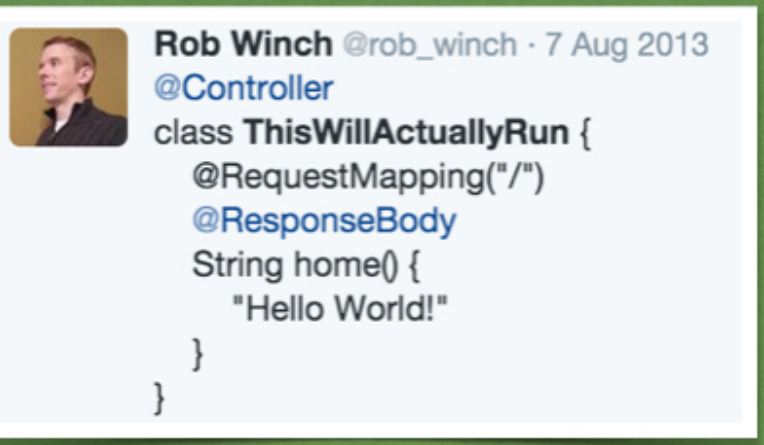
Keep services have the logic
Intelligence in the endpoints



HOW TO DEPLOY

Automatic deployments
Chef, Puppet, Ansible
Docker
Continuous delivery

MICRO SERVICE IN A TWEET



MORE SERVICE, LESS MICRO

Focus on building services that make development and deployment easier, not just tiny services

Chris Richardson

IMPLEMENTATION TIPS

- Start with less number of services and let them evolve
- Use open standards like HTTP and ATOM
- Invoker knows what a service consumes and produces
- If a service dependent to the other but not using its services, merge them
- Externalize and version all configurations
- Monitor logs from single place, like Splunk or LogStash & Kibana
- Avoid services that cannot live without the others
- Run automatic system tests regularly
- Keep track of technical debt in a separate backlog

PROS

Robust
Simple
Scalable
Testable
Composition of systems
Easy to change
Easy to replace
Easy to deploy
Language agnostic
Platform agnostic
Framework agnostic
Easy to remove legacy code!

CONS

Poor teams will always create poor systems
Adds additional complexity
You have to manage performance well
Needs trial and error
Changes require full re-deploy
Still somewhat coupled
Large codebase
Complex Versioning
Latency

REFERENCES

"Microservices" by James Lewis & Martin Fowler
http://martinfowler.com/articles/microservices.html

"Microservices are SOLID" by Matt Stine
http://www.mattstine.com/2014/06/30/microservices-are-solid/

"Micro Services Architecture" by Michael Ibarra
http://files.meetup.com/8790462/MicroServicesArch-Slide.pdf

"Microservices: Decomposing Applications for Deployability and Scalability" by Chris Richardson
http://www.slideshare.net/chris.e.richardson/developing-apps-with-a-microservice-architecture-svforum-microservices-meetup

"Micro-Services - Java, the UNIX way" by James Lewis
http://vimeo.com/74452550

"SOA Manifesto" translated into Turkish by Lemi Orhan Ergin
http://www.soa-manifesto.org/default_turkish.html

IMAGES

"Sleeping Ladybug" by Robert Körner
https://www.flickr.com/photos/67611651@N03/8741277037

"Ward Cunningham" on Wikimedia Commons
http://commons.wikimedia.org/wiki/File:Ward_Cunningham_-_Commons-1.jpg

"Godzilla Simpsons" on Wikimedia Commons
http://commons.wikimedia.org/wiki/File:Godzilla_Simpsons.svg

"Martin Fowler" on Hürriyet Gazetesi
http://www.hurriyet.com.tr/ekonomi/27166072.asp

"Michael Ibarra" on Twitter
https://twitter.com/bm2yogi

"Chris Richardson" on Github
https://github.com/cer

"Melvin E. Conway" on FlowChainSensei
http://flowchainsensei.wordpress.com/2012/09/07/conways-law-revisited

All icons and drawings by Freepik
http://www.flaticon.com

LEMİ ORHAN ERGİN

AGILE SOFTWARE CRAFTSMAN

lemiorhan@agilistanbul.com
Founder & Author @ agilistanbul.com

@LEMIORHAN
https://www.linkedin.com/in/lemiorhan

@LEMIORHAN
https://twitter.com/lemiorhan

@LEMIORHAN
http://www.slideshare.net/lemiorhan

@LEMIORHAN
https://github.com/lemiorhan

AGILISTANBUL.COM
Turkish blog about agile development

LEMIORHANERGIN.COM
Official site having personal information