EVOLUTIONARY (SOFTWARE) ARCHITECTURES

wintech italia

PAOLO ROSSI
DELPHI DEV

paolo@paolorossi.net
www.paolorossi.net
blog.paolorossi.net

DelphiDay
italian conference
PIACENZA EDITION

wintech italia

AGENDA

- → Architectures
- → Evolutionary Architectures
- → Change and Architectural Dimensions
- → Fitness Functions
- → Evolving DataBases
- → Examples and Tools
- → Books & Resources

SOFTWARE ARCHITECTURES

The scope is too large for a concise definition

Ralph Johnson's definition

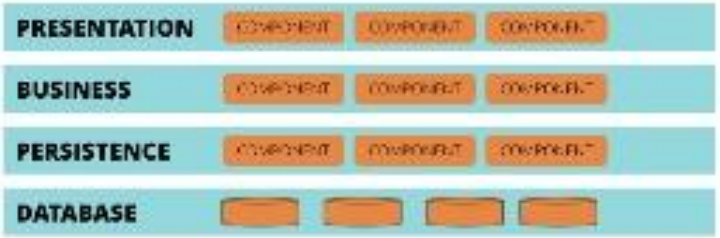
the important stuff (whatever that is)

SOFTWARE ARCHITECTURES

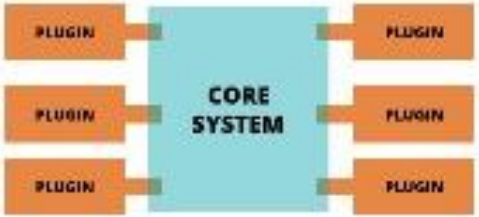
Hard to change over time

Example Architectural Patterns

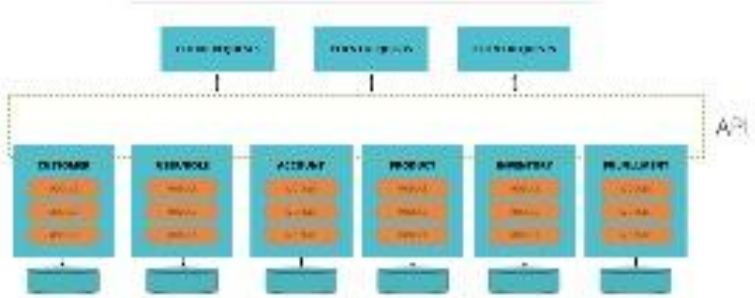
Big Ball of Mud



Layered Architecture



Microkernel



Microservices

EVOLUTIONARY ARCHITECTURES

"An evolutionary architecture supports guided, incremental change across multiple dimensions"

ARCHITECTURAL FACTORS

➔ Architects must determine the most important ones

- Accessibility
- Scalability
- Performance
- Security
- etc...

"ILITIES"

accessibility
accountability
accuracy
adaptability
administrability
affordability
agility
auditability
autonomy
availability
compatibility
composability
configurability
correctness
credibility
customizability
debugability
degradability
determinability
demonstrability
dependability
deployability
discoverability
distributability
durability
effectiveness
efficiency
usability
extensibility
transparency
fault tolerance
fidelity
flexibility
inspectability
installability
integrity
interoperability
learnability
maintainability
manageability
mobility
modifiability
modularity
operability
orthogonality
portability
precision
predictability
producibility
provability
recoverability
relevance
reliability
repeatability
reproducibility
resilience
responsiveness
reusability
robustness
safety
scalability
seamlessness
self-sustainability
serviceability
securability
simplicity
stability
standards
compliance
survivability
sustainability
tailorability
testability
timeliness
traceability

TIME AS 4D

We want to add a new standard “-ility” to software architecture

EVOLVABILITY

The time becomes the 4th dimension

EVOLUTIONARY ARCHITECTURES

Software becomes harder to change over time

Don’t let that become a self fulfilling prophecy

CHANGE

IS INEVITABLE

TECHNICAL

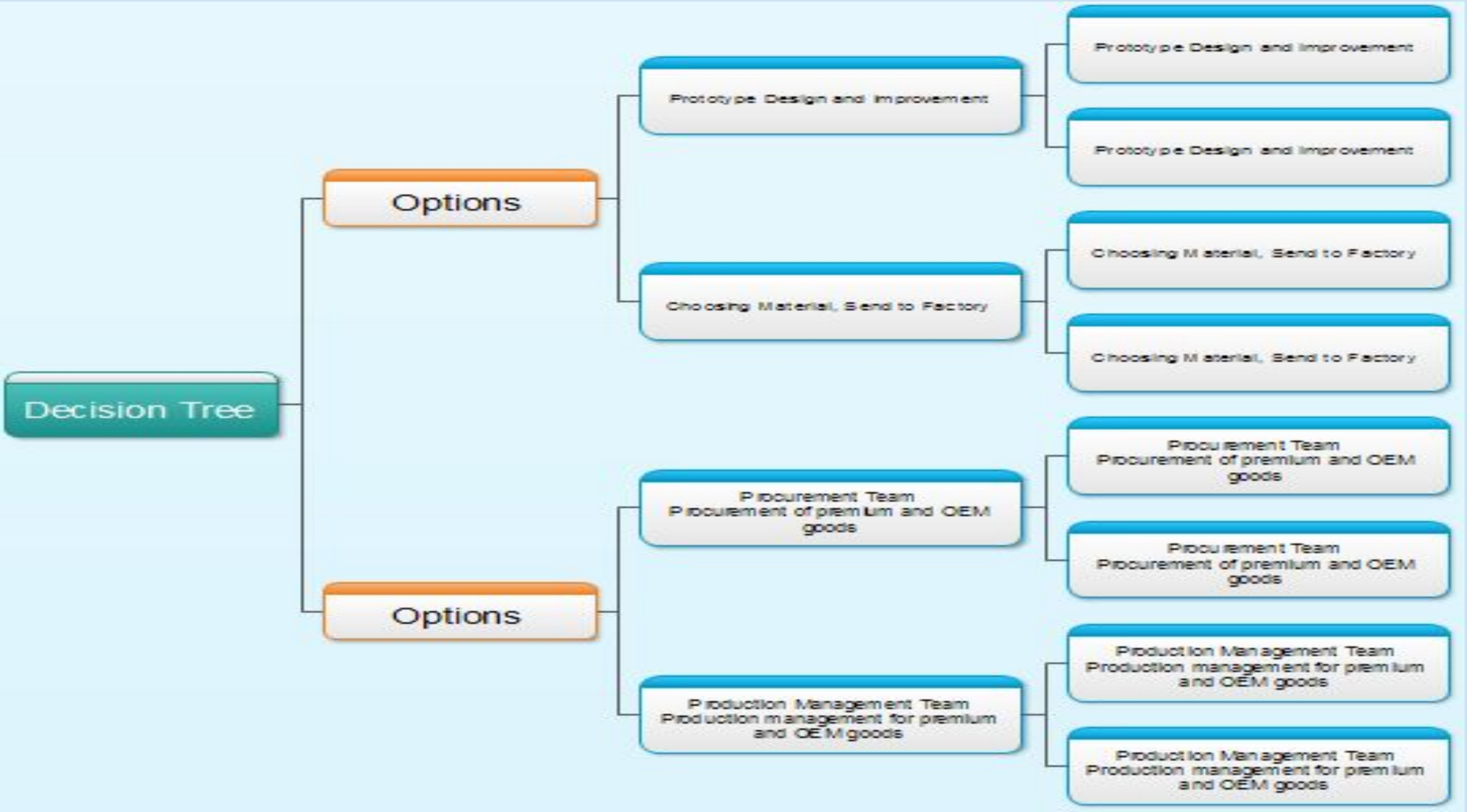
- → Programming languages
- → Programming language evolution
- → Libraries
- → Frameworks
- → Tools
- → Constraints

DOMAIN

- → Revenue models
- → Competitors
- → Customer needs
- → Markets
- → Products
- → Legal requirements

IF CHANGE

IS INEVITABLE THEN


Decision Tree
Options
Prototype Design and Improvement
Prototype Design and Improvement
Prototype Design and Improvement
Choosing Material, Send to Factory
Choosing Material, Send to Factory
Choosing Material, Send to Factory
Options
Procurement Team
Procurement of premium and OEM goods
Procurement Team
Procurement of premium and OEM goods
Procurement Team
Procurement of premium and OEM goods
Production Management Team
Production management for premium and OEM goods
Production Management Team
Production management for premium and OEM goods
Production Management Team
Production management for premium and OEM goods

INCREMENTAL CHANGE

- → How teams build software incrementally
- → How they deploy it
- → Continuous delivery is a key practice

GUIDED CHANGE

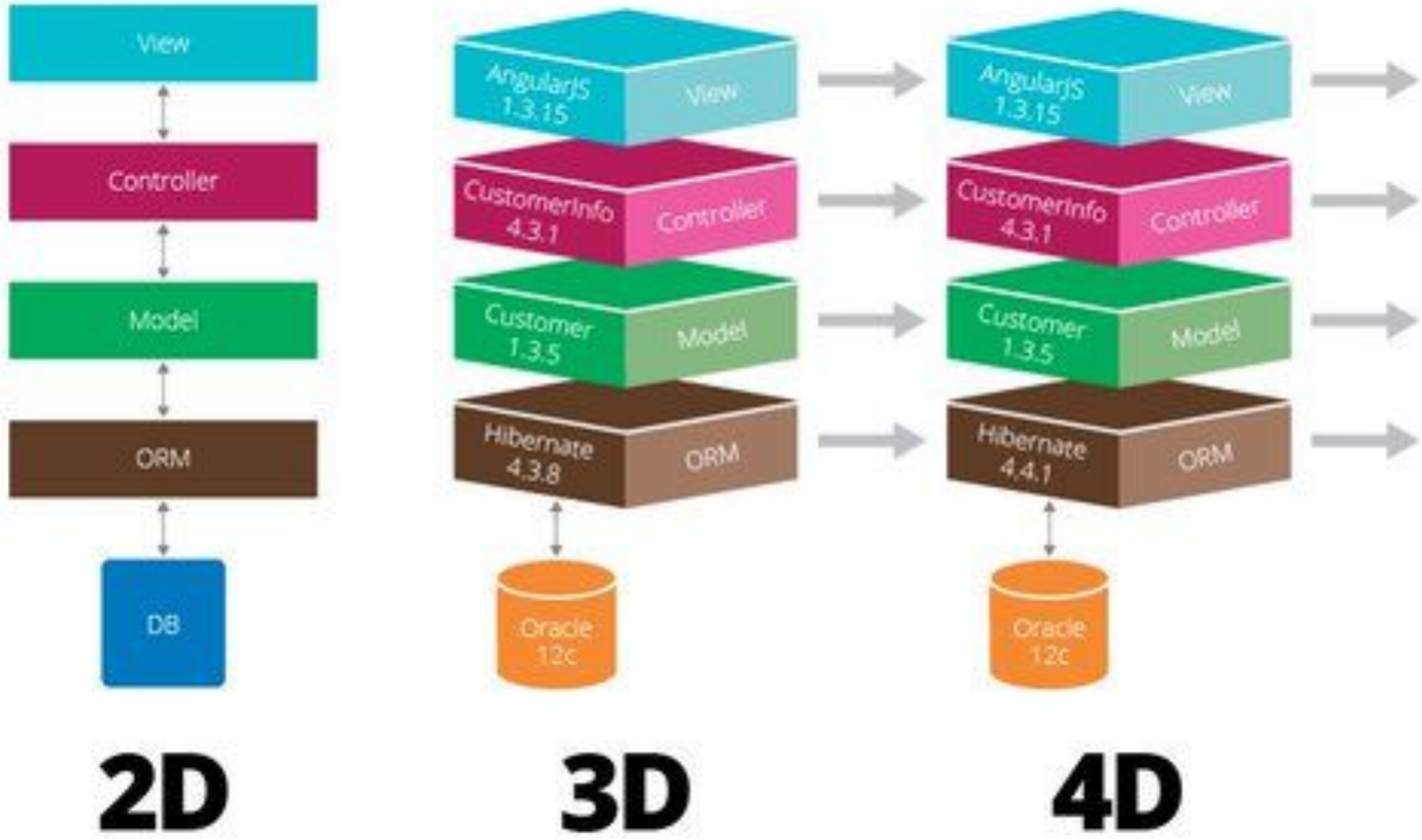
- We want to guide the change
  - Rather than suffer from it
- Introducing the “Fitness Functions”
  - From evolutionary computing
- The key is to measure everything
  - More on later

ARCHITECTURAL DIMENSIONS

There are no separate systems

ARCHITECTURAL DIMENSIONS

- → Technical dimensions
  - ◆ Languages
  - ◆ Frameworks
  - ◆ Libraries
- → Beyond technical architecture
  - ◆ Legal aspects
  - ◆ Auditability


View
Controller
Model
ORM
DB
AngularJS 1.3.15
View
CustomerInfo 4.3.1
Controller
Customer 1.3.5
Model
Hibernate 4.3.8
ORM
Oracle 12c
AngularJS 1.3.15
View
CustomerInfo 4.3.1
Controller
Customer 1.3.5
Model
Hibernate 4.4.1
ORM
Oracle 12c
2D
3D
4D

CONWAY'S LAW

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

ARCHITECTURAL TEAMS

- → The social structures (the communication paths between people) inevitably influence final product design
- → Delegation means communication problems (or difficulties)

ARCHITECTURAL TEAMS

→ Example:

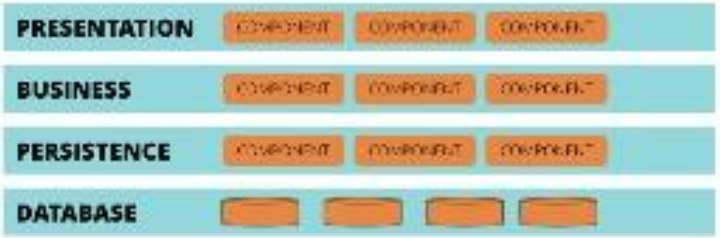
- Layered architecture: team is separated by technical function (UI, business logic, data access)
- Solving common problems that cut vertically across layers **increases** coordination work

ARCHITECTURAL TEAMS

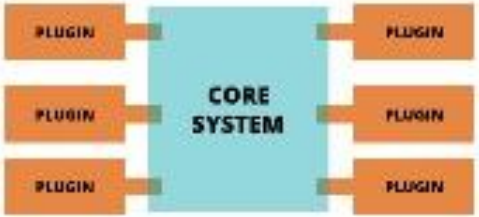
Structure teams like your target architecture

Example Architectural Patterns

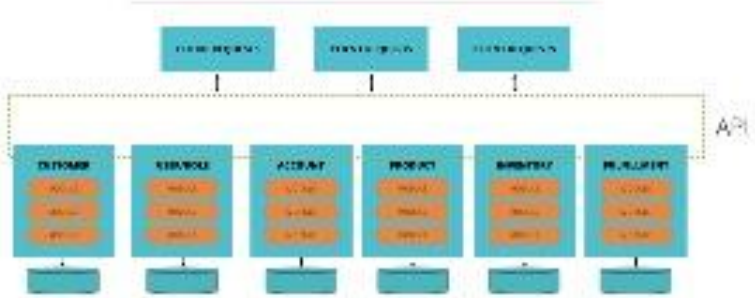
Big Ball of Mud



Layered Architecture



Microkernel



Microservices

FITNESS FUNCTIONS

DEFINITION

"An architectural **fitness function** provides an objective integrity assessment of some architectural characteristic(s)"

F.F.

- ➔ Fitness functions check that developers preserve important architectural characteristics
- ➔ But, what is better?
- ➔ Find a way to measure better
- ➔ Ok, but what is a FF in reality?

F.F.

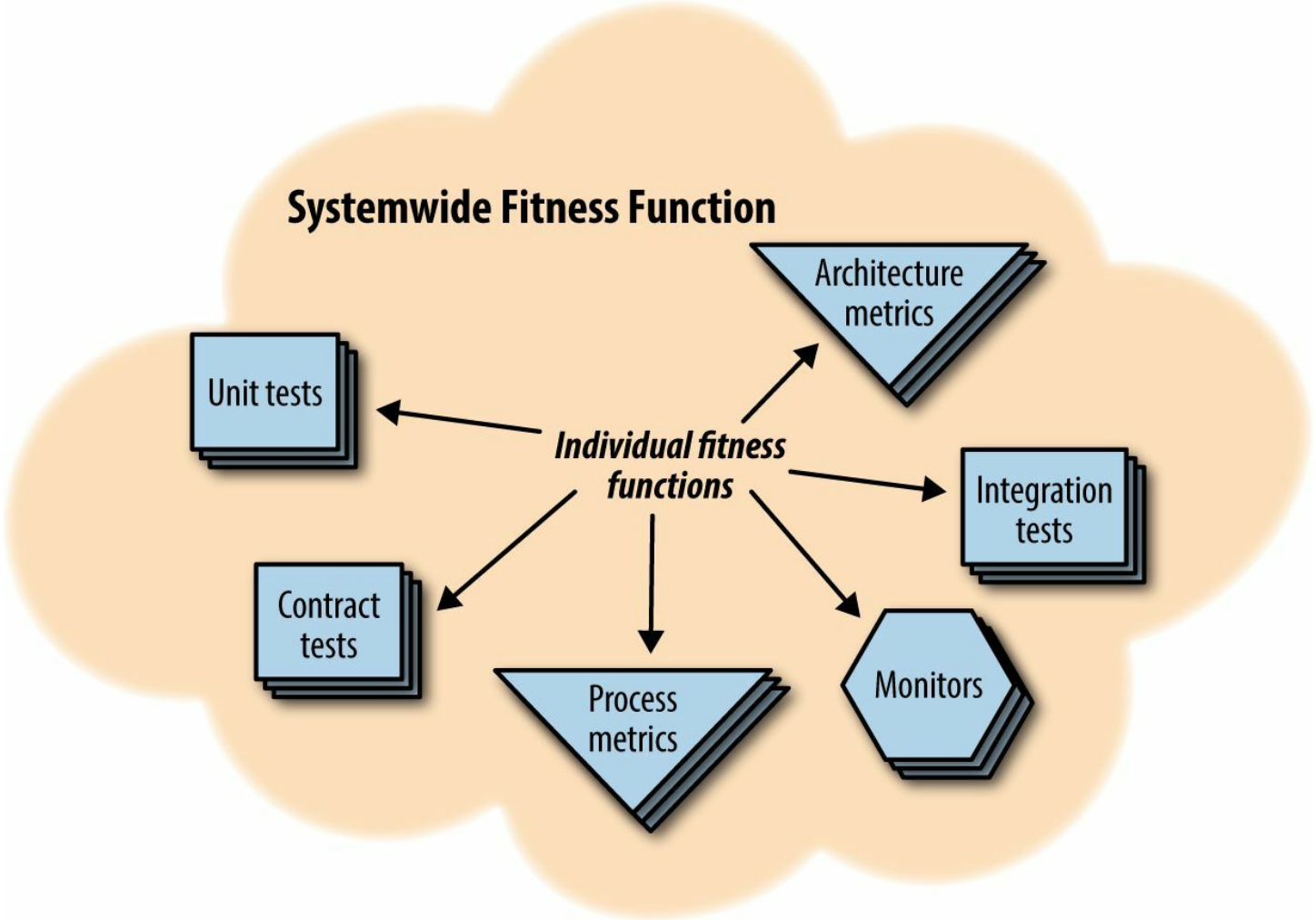
- Specific architectural requirements differ greatly across systems and organizations
- They are based on
  - Business requirements
  - Technical capabilities
  - Client needs
  - ...

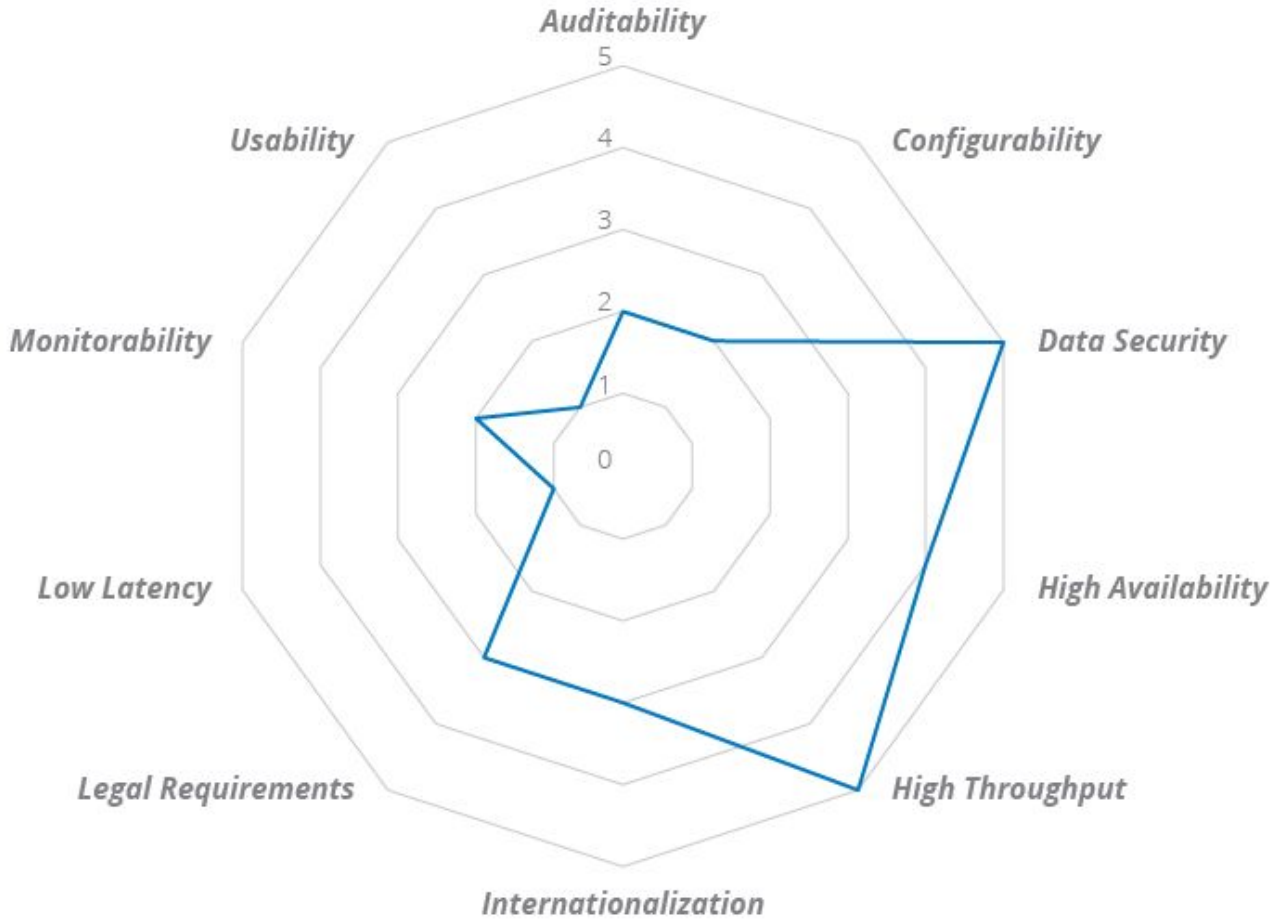
F.F.

- Examples of FF:
  - Intense security
  - Low latency
  - Resilience to failure
- Remember the “-ilities”
- Fitness functions embody a protection mechanism for the “-ilities” of a given system

SYSTEMWIDE F.F.

- → Collection of FF
- → They help to “measure” the system as a whole
- → There are **tradeoff**
  - ◆ Is more important the scalability or the security?
  - ◆ So


Systemwide Fitness Function
Architecture metrics
Unit tests
Individual fitness functions
Integration tests
Contract tests
Process metrics
Monitors


Auditability
5
4
3
2
1
0
Configurability
Data Security
High Availability
High Throughput
Internationalization
Legal Requirements
Low Latency
Monitorability
Usability

REAL WORLD EXAMPLES

- Performance
  - Server have to respond in 100ms
- Scalability
  - System must manage up to 100.000 sessions
- Coding standard
  - Cyclomatic complexity must be lower than 100
- Legal requirements
  - GDPR must be complied with

REAL WORLD EXAMPLES

- → At every iteration we know how if the system remains closer the the goals
- → Save the FF and look at them over time
- → Introduce FF early (and often) to pick up inflection points
- → Measure everything

TOOLS

- Version control systems
  - Git, Subversion
- Continuous delivery
  - Jenkins, Travis
- Code Review
  - Delphi Metrics, Pascal Analyzer, Scientific Toolworks Understand

TOOLS

- Configuration Management
  - Chef, Puppet
- Monitoring
  - Nagios, Zabbix
- Containers
  - Docker, Kubernetes

DATABASE

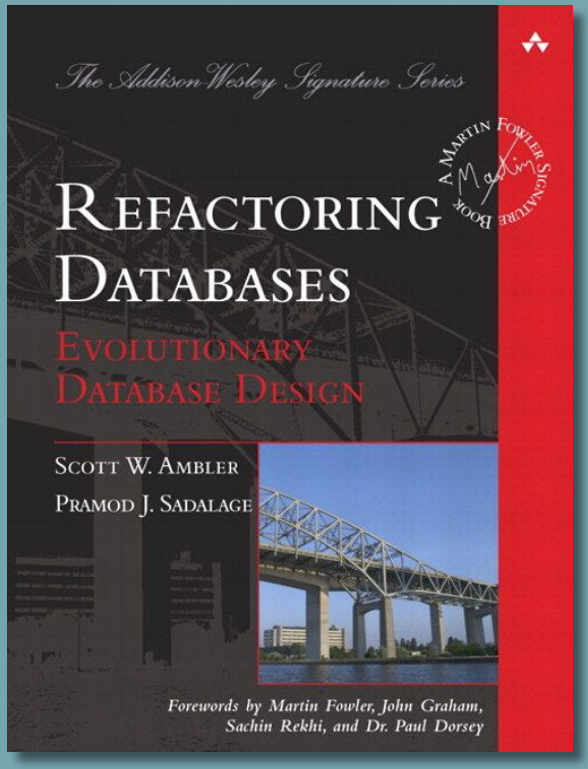
Evolution?????

EDBD

- **E**volutionary **D**ata**B**ase **D**esign
  - Aka Database refactoring
- Controversial topic
- Software evolves
  - Why not the schema data?
  - Why not the Database
- EDBD is not about change in the schema data only as consequence
- EDBD is about embracing the change (evolution)

EDBD

The bible



TDBD

- → **T**raditional **D**ata**B**ase **D**esign
- → TDBD assumes that evolving database schemas it's a hard thing to do, so is best not do do it
- → TDBD needs to model in detail all data aspects of a system early in the life-cycle
- → Nothing wrong with that but this is not the reality
- → What's the average state of real databases?

TDBD

- Existing state of production databases
  - Columns that are no longer being used
  - Columns never used
  - Columns that are being used for several purposes
  - Columns that contains various data type
    - Usually string columns containing other types

TDBD

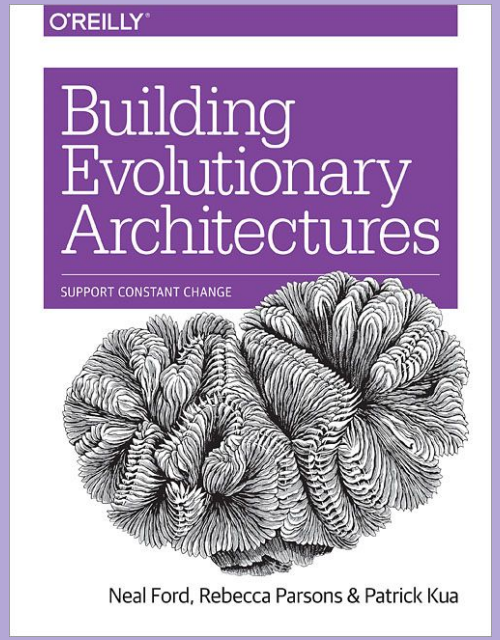
- → In practice we are not very good at getting the design right up front
- → So the requirements do in fact change over time

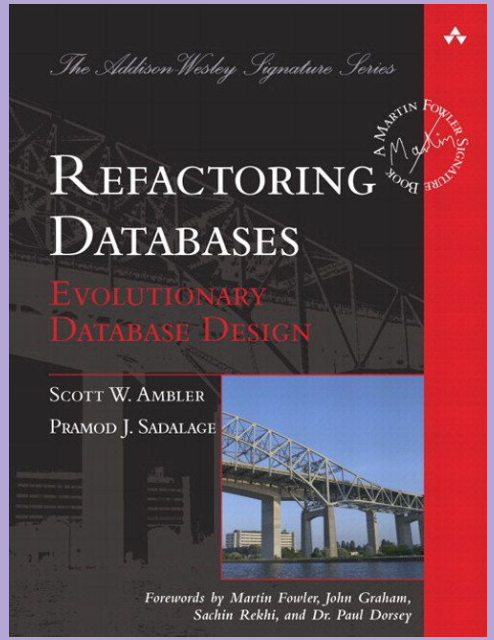
In a fast changing environment TDBD is **not the right approach**

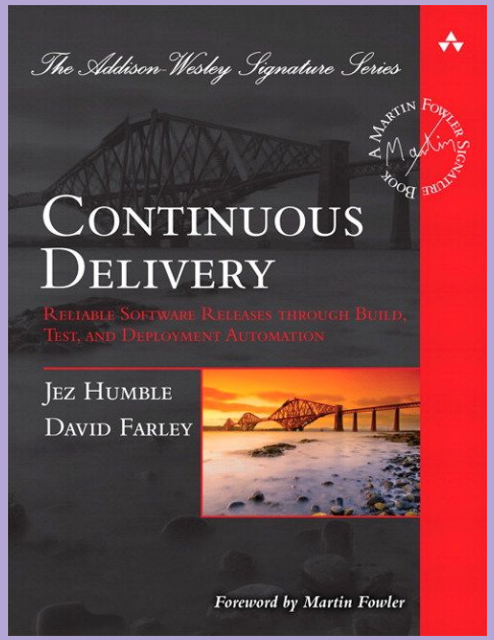
EDBD VS TDBD

- Resistance among traditional DBA and DB designers
  - Too much DB sandboxes
  - Duplicate data management
  - Temporary time-frame in which we have multiple schema alive
  - Triggers to ensure data integrity

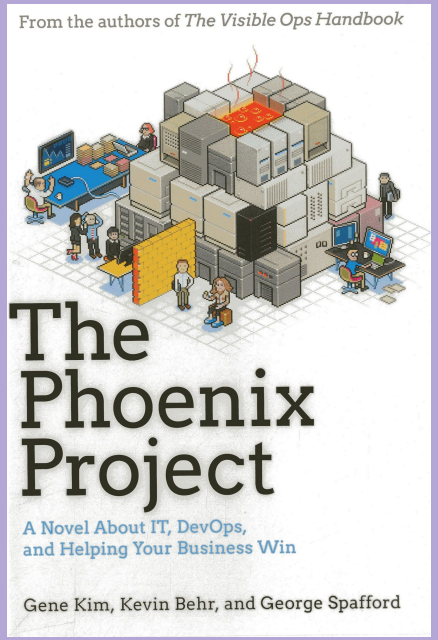
BOOKS

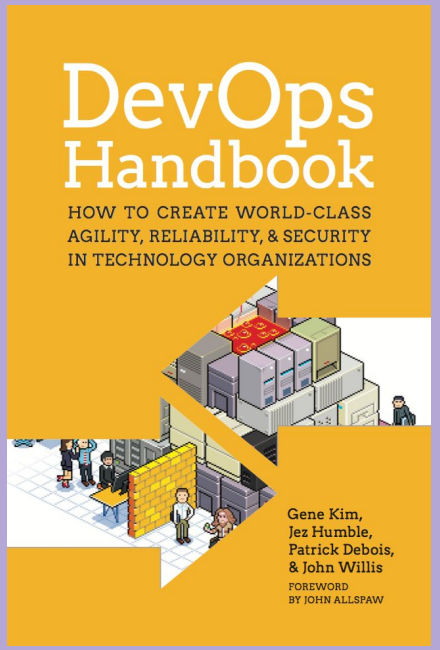

O'REILLY
Building Evolutionary Architectures
SUPPORT CONSTANT CHANGE
Neal Ford, Rebecca Parsons & Patrick Kua


The Addison Wesley Signature Series
A Martin Fowler Signature Book
Refactoring Databases
Evolutionary Database Design
Scott W. Ambler
Pramod J. Sadalage
Forewords by Martin Fowler, John Graham, Sachin Rekhi, and Dr. Paul Dorsey


The Addison Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler

BOOKS


From the authors of *The Visible Ops Handbook*
The Phoenix Project
A Novel About IT, DevOps, and Helping Your Business Win
Gene Kim, Kevin Behr, and George Spafford


DevOps Handbook
HOW TO CREATE WORLD-CLASS AGILITY, RELIABILITY, & SECURITY IN TECHNOLOGY ORGANIZATIONS
Gene Kim, Jez Humble, Patrick Debois, & John Willis
FOREWORD BY JOHN ALLSPAW

ONLINE RESOURCES

- Microservices as an Evolutionary Architecture:
  - https://www.thoughtworks.com/insights/blog/microservices-evolutionary-architecture
- Characteristics of Evolutionary Architectures:
  - https://www.infoq.com/news/2016/03/evolutionary-architectures
- My take on Evolutionary Architecture:
  - https://medium.com/developers-writing/my-take-on-evolutionary-architecture-f761d45e75b9
- Approaches to Evolutionary Architectural Design:
  - http://ncra.ucd.ie/papers/JonathanByrneThesis.pdf
- Evolutionary Database Design
  - https://www.martinfowler.com/articles/evodb.html


That's all Folks!