EQUINOX IT

From legacy to the future – *Can we get there from here?*

… and do we want to?

Bill Ross
Principal Consultant
Equinox IT

Legacy system - *where are we?*

- Core to your business
- Systems of record
- Stable
- Many integrations
- Completely depreciated

- Large
- Complex
- Hard to change
- Expensive to run
- Limited expertise

Monolithic

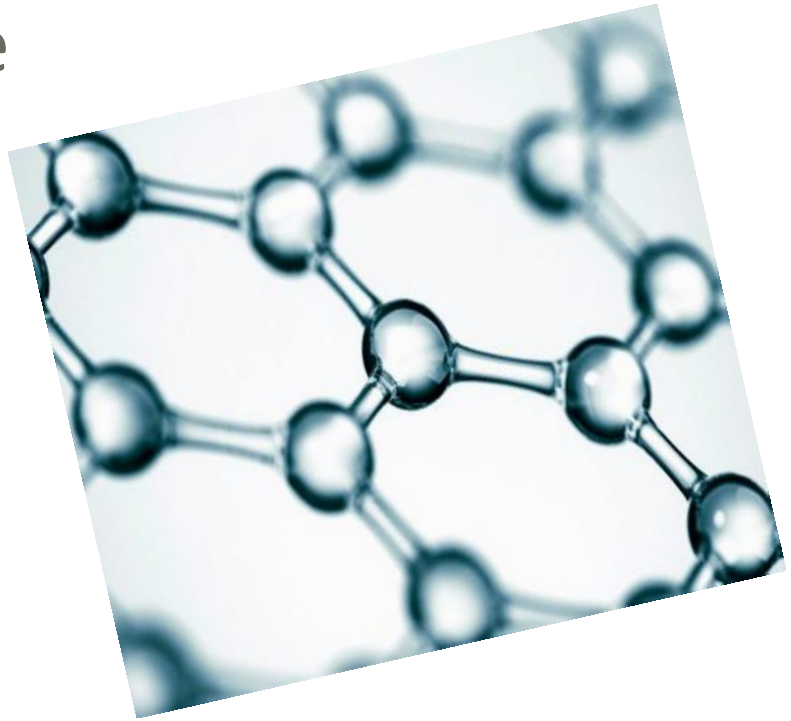
"big-ball-of-mud" architecture

Monoliths can be modular and maintainable

- Layered architecture
- Modules with clear responsibilities
- Dependencies on interfaces implementations
- Support complex domain models

The future - *where (we think) we want to be?*

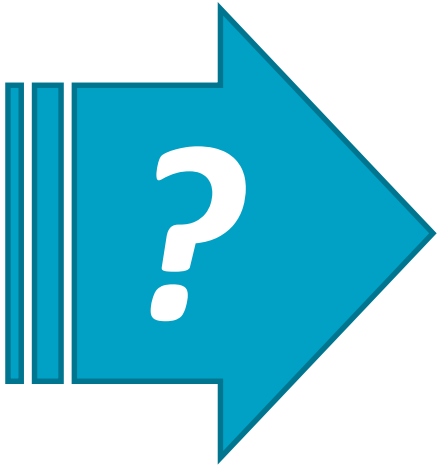
- Microservices architecture
- Event driven
- Polyglot development languages
- Polyglot storage

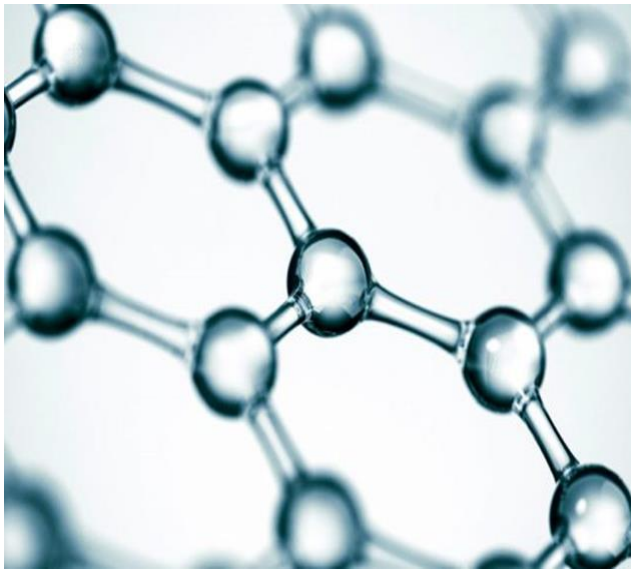


What is a microservices architecture?

- Uses services as the unit of modularity
- Each service corresponds for a business capability
- Each service is independently maintained
- Each service can use the most different development language and storage technology

Can we get there from here?

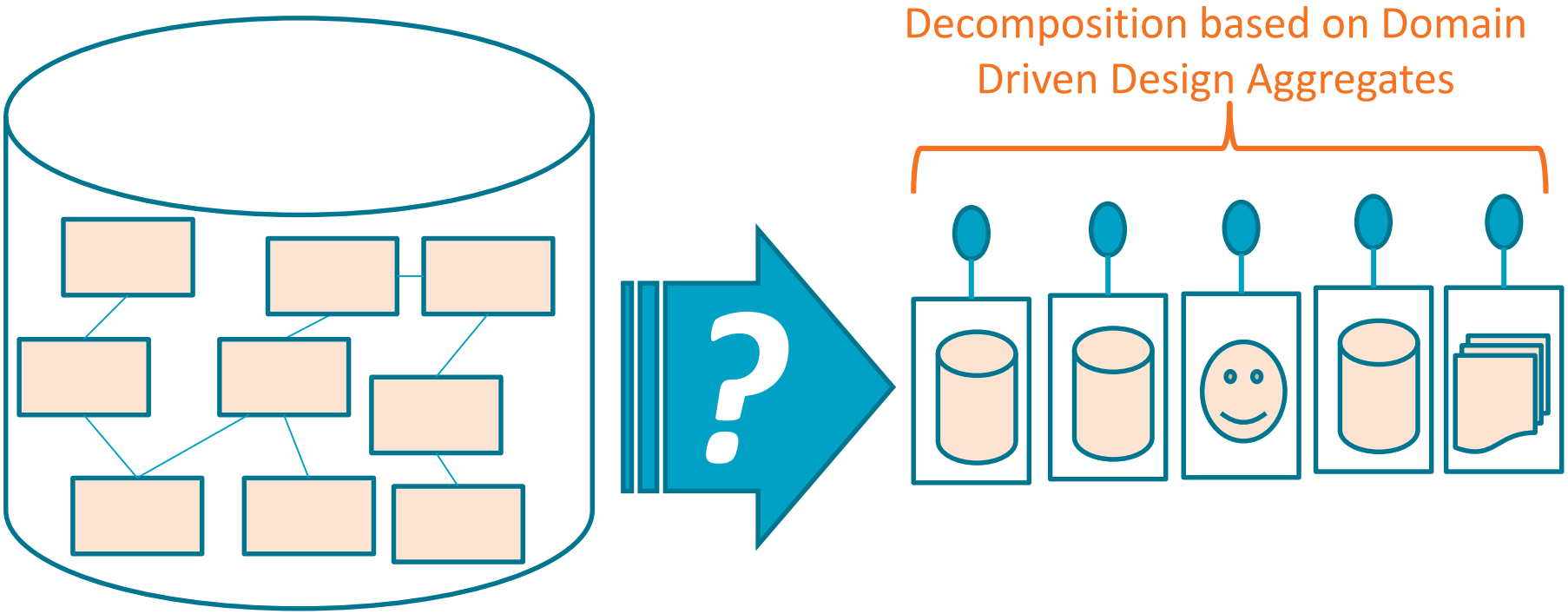




… and do we want to?

EQUINOX IT

Decomposing a domain model



Monolithic Database

- Single storage technology
  Enforced relationships
- Controlled and versioned as one unit

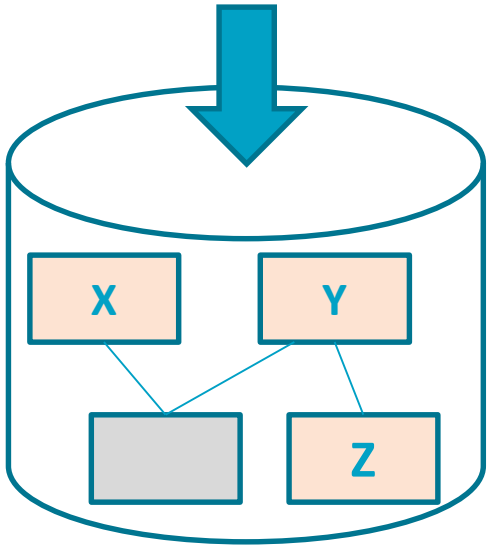
Microservices Datastores

- Multiple storage technologies
- Client controlled relationships
- Controlled and versioned independently

Transactions that span services

Begin Transaction
- Update X
- Update Y
- Update Z

Success – Commit
Error - Rollback



Monolithic Database

- Update X

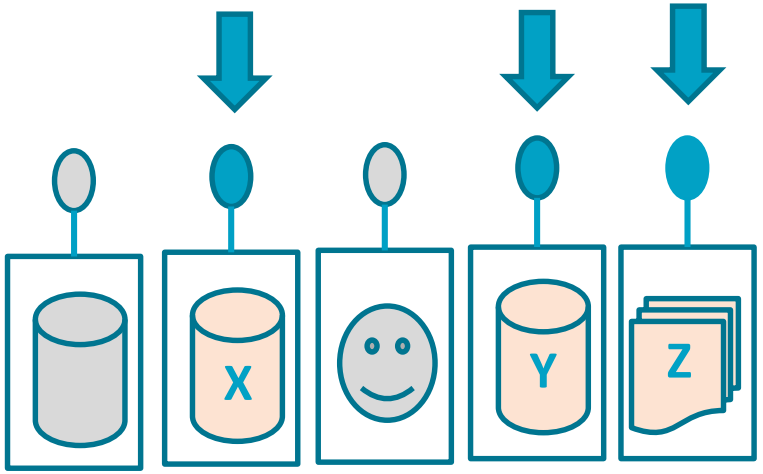
Did it work?
- Update Y

Did it work?

Error – Compensate (undo X)
- Update Z

Did it work?

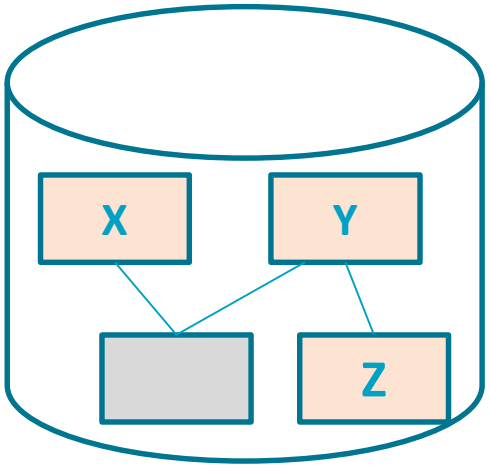
Error Compensate (undo X and Y)



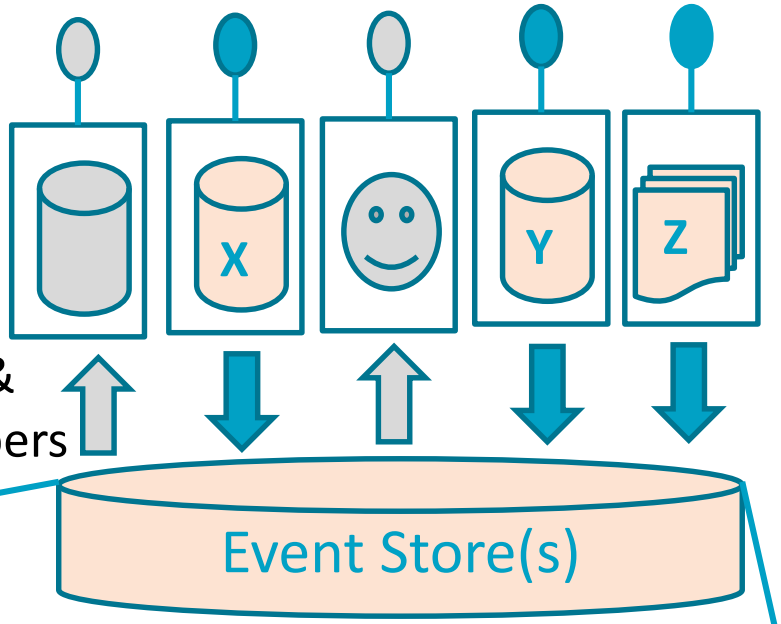
Microservices

Event sourcing - *eventual consistency*

Monolithic Database



Microservices



Event_id	Type	Entity_ type	Entity_id	Event_data
1001	Loan Requested	Loan	111	{...}
1002	Loan Applicant Validated	Loan	111	{...}
1003	Credit Check Completed	Loan	111	{...}
1004	Loan Approved	Loan	111	{...}
...	...	...	...	{...}

EQUINOX IT

Event Store vs. RDBMS

"The truth is the log [event store].

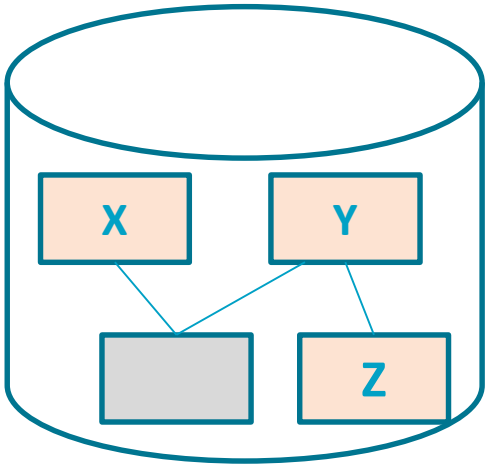
The database is a cache of a subset of the log."

- Pat Helland

The relational database management systems use a structure and language consistent with first order predicate logic.

Querying and reporting

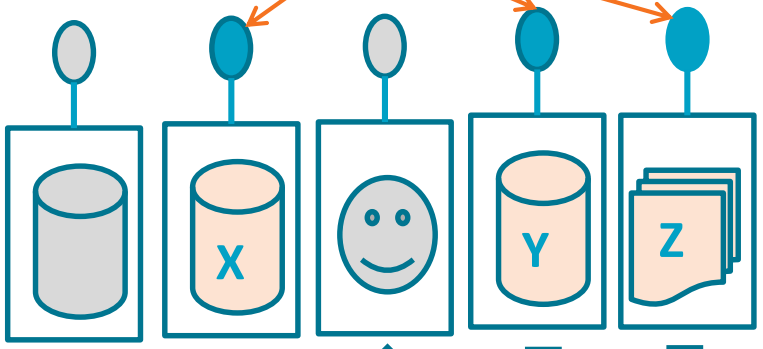
Monolithic Database



Select * From X, Y, Z
Where X.id = Y.xid and
Y.Value > 1000 and
Y.Id = Z.yid

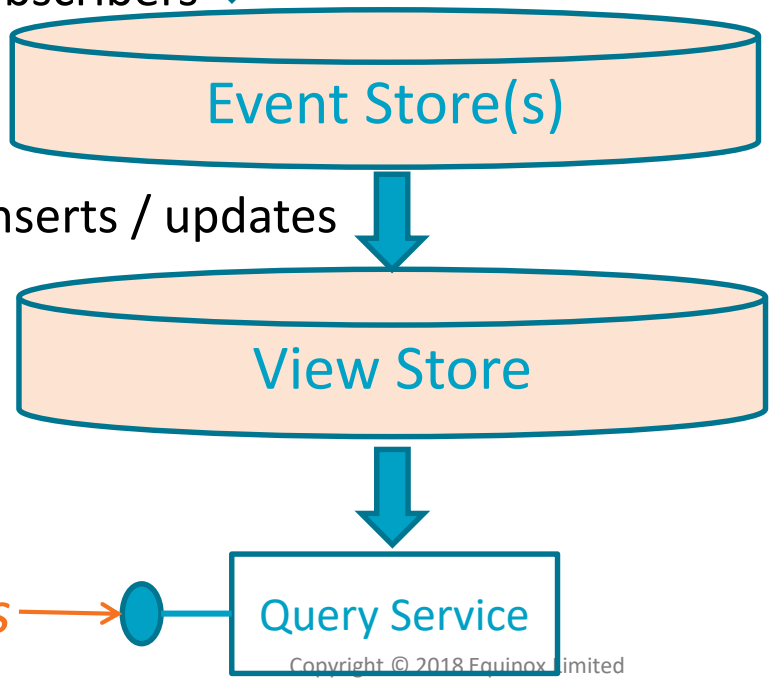
Commands

Microservices



events &
subscribers

Command Query Responsibility Separation



EQUINOX IT

Modularity for maintainability

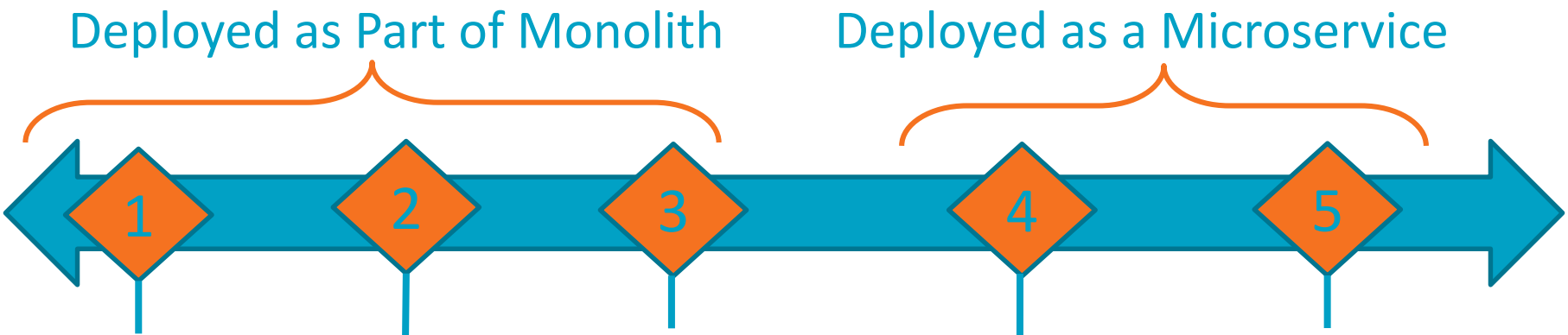
What is it?

- Separation of the application code into logical units that have specific responsibilities
  - Separation of concerns
  - Single responsibility principle

Why do it?

- Improves understandability, and therefore supports:
  - Maintainability
  - Enhanceability
  - Longevity

Feature Packaging & Deployment Spectrum



Namespace
Single Deployment

Feature in Module
Single Deployment

Feature in Module
Separate Package
Separate Versioning

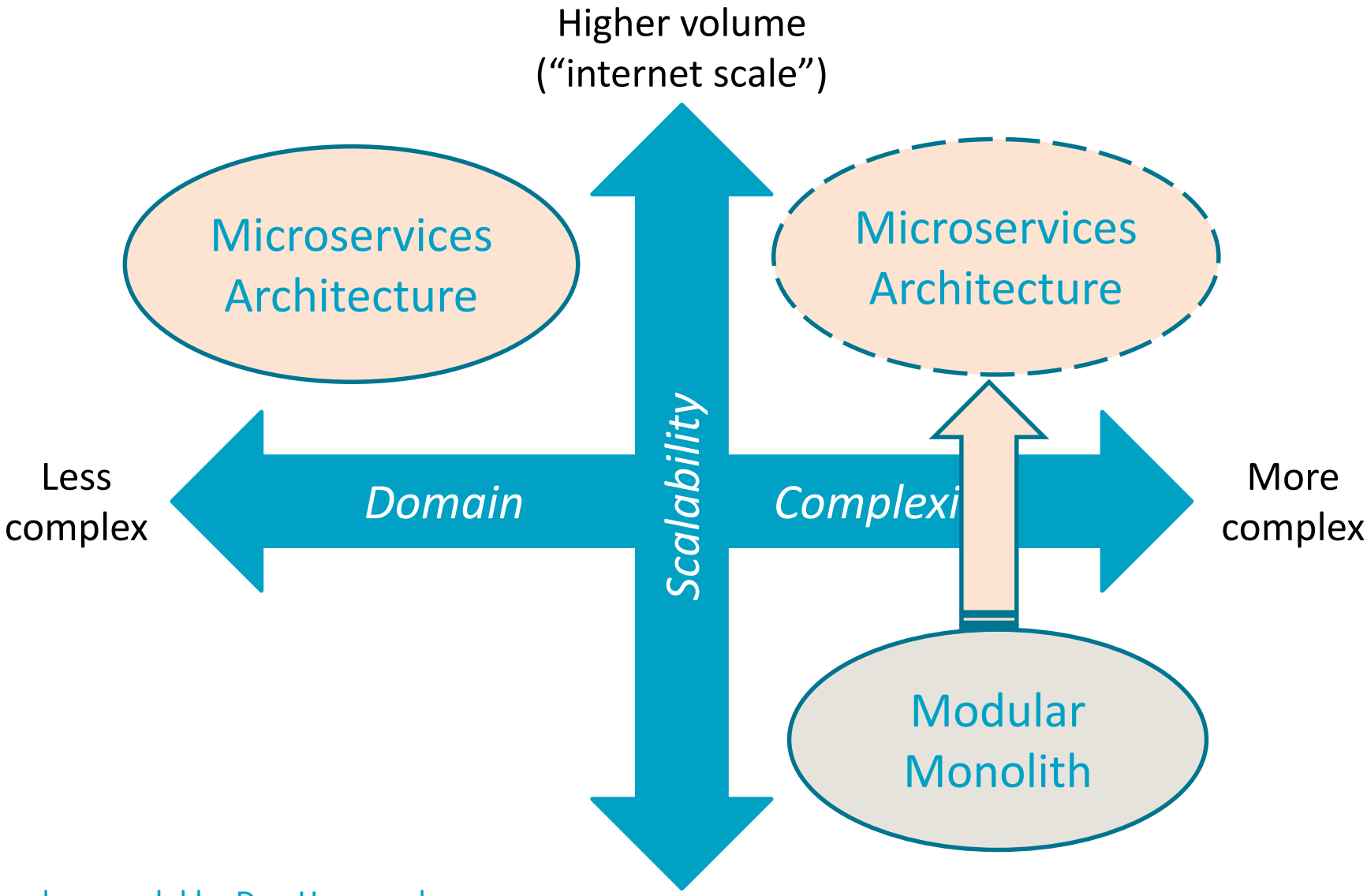
Feature in Module
Separate Package
Separate Versioning
Separate Process
Accessed Synchronously

Feature in Module
Separate Package
Separate Versioning
Separate Process
Accessed Synchronously

Based on model by Dan Haywood

EQUINOX IT

Scalability vs. Doman Complexity



Based on model by Dan Haywood

EQUINOX IT

Some thoughts…

First Law of Distributed Object Design: "don't distribute your objects"

- Martin Fowler

If you can’t manage building a monolith inside a single process, what makes you think putting network in the middle is going to help?

- Greg Young

Reasons to go to microservices architecture

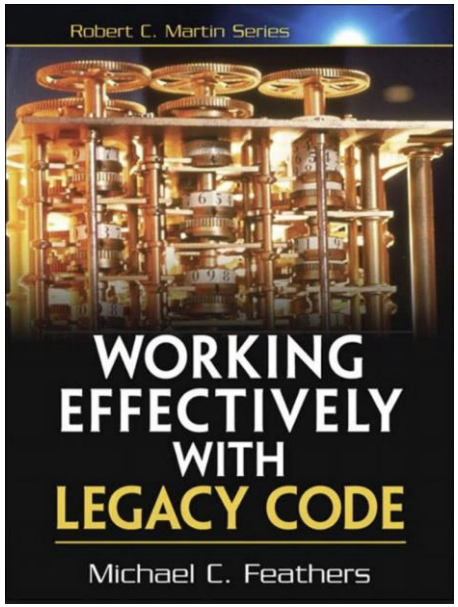
- Scalability
- Availability
- Flexibility
- Productivity
- Maintainability?
- Modernisation??
- Fashion???

- ~~Performance~~
- ~~Reduce complexity~~
- ~~Reduce cost~~
- ~~Complex domain models~~

Legacy code

“Legacy code is code that doesn’t have unit test”

- Michael C. Feathers, Working Effectively with Legacy Code, 2005.



The Legacy Code Dilemma

When we change code, we should have tests in place. To put tests in place , we often have to change code.

Don’t get fixated on the latest plumbing

Resources

Microservices and the First Law of Distributed Objects
https://www.martinfowler.com/articles/distributed-objects-microservices.html

Event Sourcing
https://martinfowler.com/eaaDev/EventSourcing.html

CQRS https://martinfowler.com/bliki/CQRS.html

- Martin Fowler

Working Effectively with Legacy Code, 2005

- Michael C. Feathers

Building Microservices

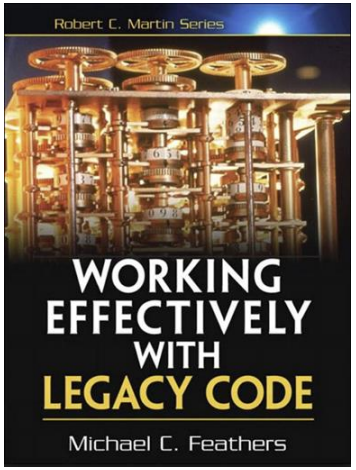
- Sam Newman

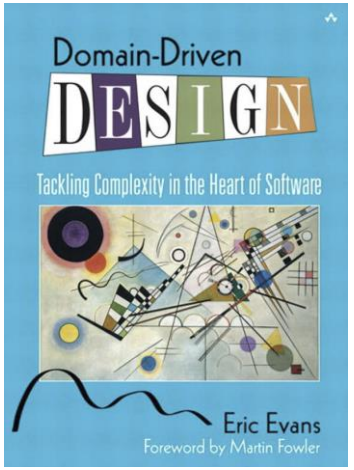
Domain Driven Design

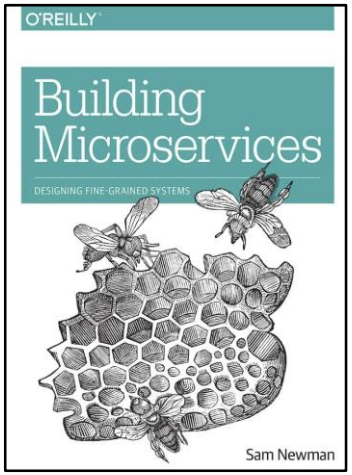
- Eric Evans

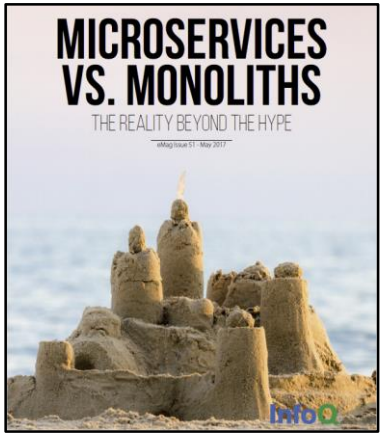
Microservices vs. Monoliths

- InfoQ eMag









Injecting fresh thinking to solve tough business problems.

EQUINOX IT