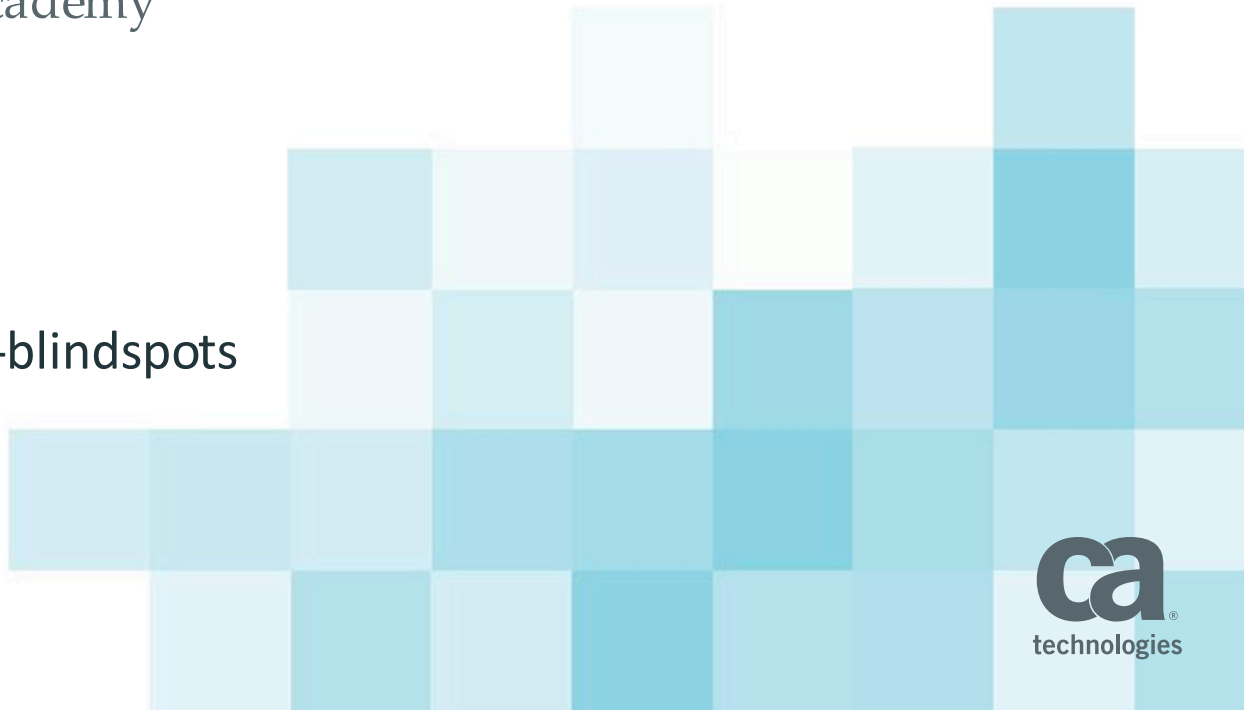
Microservices Architecture: The Blind Spots

Irakli Nadareishvili,

Director of Strategy, API Academy

CA Technologies

Deck:
http://bit.ly/microservices-blindspots



#1

What Drives Microservices Adoption?

¯_(ツ)_/¯

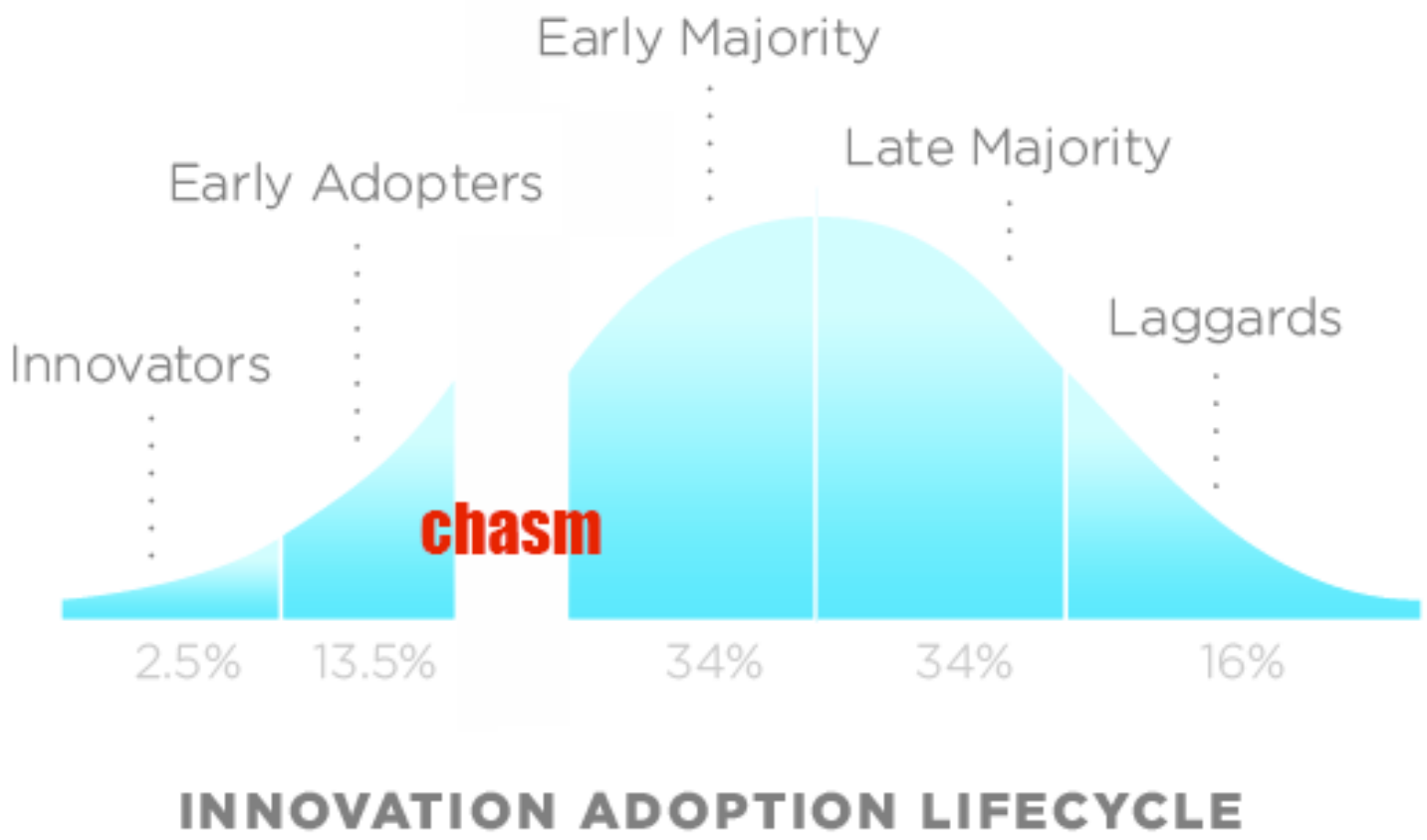

Revolution is Coming

Key Benefits of Microservices (as of 2015)

1. Technology Heterogeneity
2. Partitioned Scalability (per microservice)
3. Independent Deployments
4. Compose-ability
5. Optimized for Replace-ability

Motivations for Adoption Will Change

The reasons why MSA became popular are not the same reasons why it will become ubiquitous



Source: Crossing the Chasm – Geoffrey A. Moore

Docker is a "gateway drug" for MSA

Prediction:

Containerization will drive MSA Adoption

to the extent that:

Microservices won't be a "choice"

#2

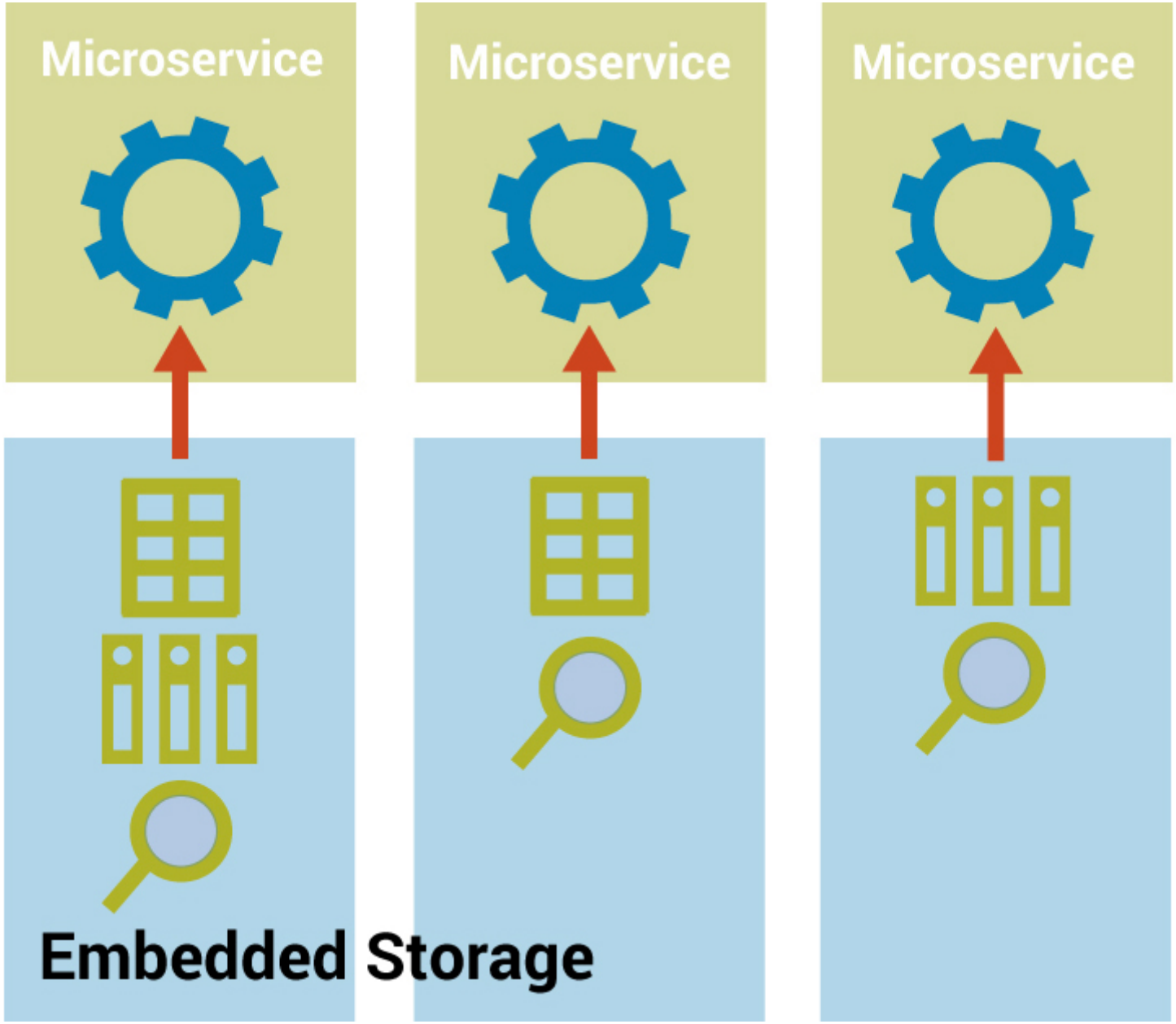
Current Focus Is Too Code-Centric

Caution: Coupling of service interfaces is every bit as toxic as the code one, and can render entire architecture useless.

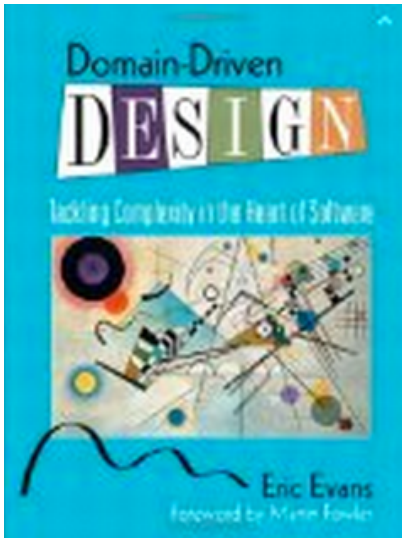
Hypermedia design, with its high degree of interface de-coupling can be of huge help here.

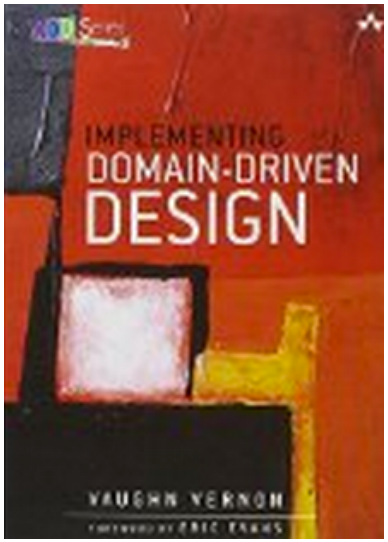
#3

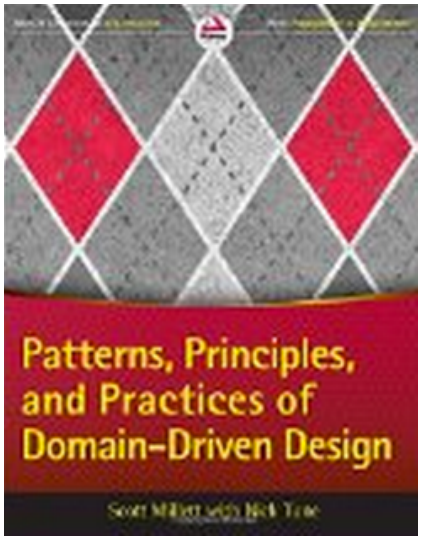
Decentralized Data



DDD & Bounded Contexts







Bounded Context = Capabilities.

"In your organization, you should be thinking not in terms of data that is shared, but about the capabilities those contexts provide the rest of the domain."

– Sam Newman, Building Microservices


How small should
a Microservice be?

“Bounded context should be as big as it needs to be in order to fully express its complete Ubiquitous Language”

– Vaughn Vernon, *Implementing Domain –Driven Design.*

Inherent Conflict:

Business teams actually don't favor small bounded contexts (due to diff. ubiquitous language).

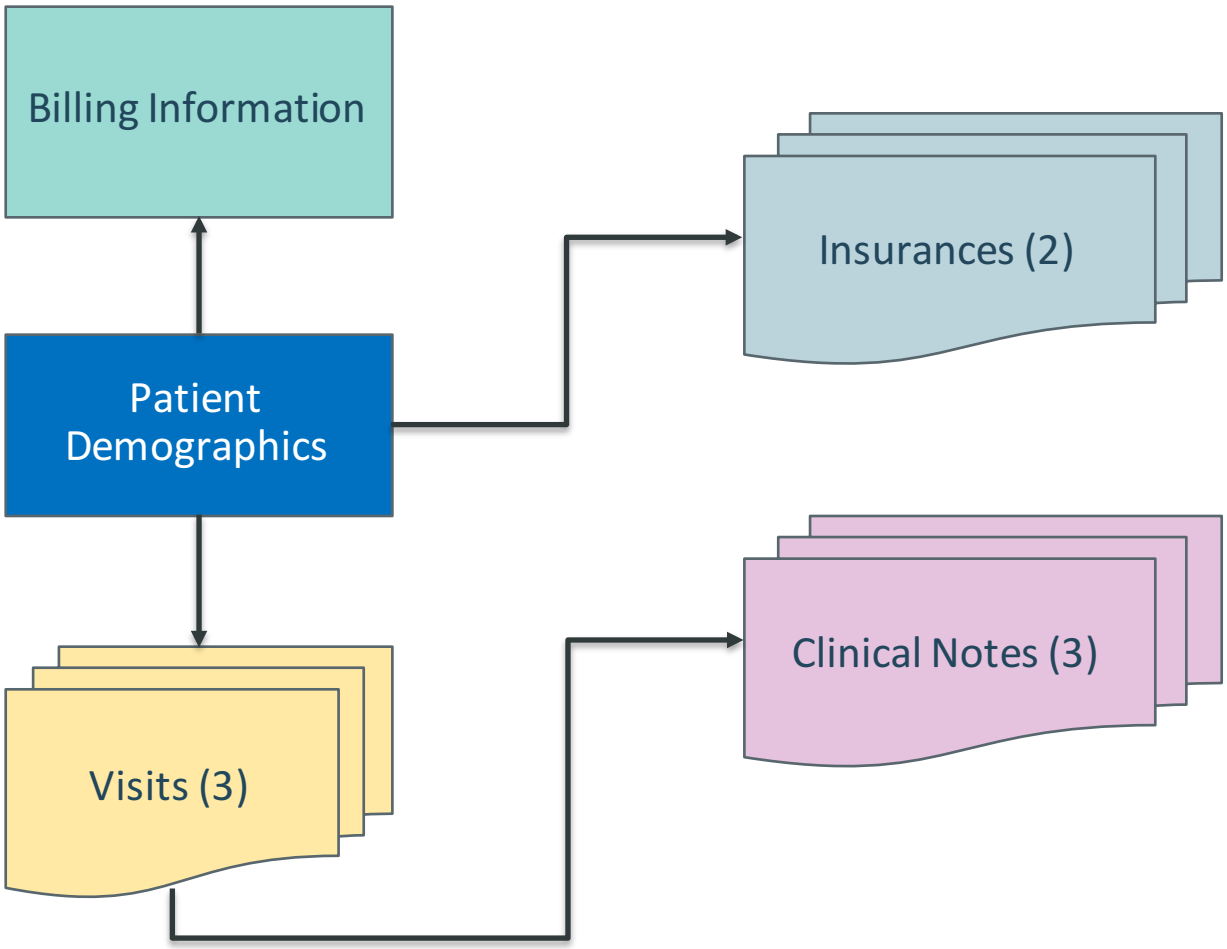
For tech: continuous Integration is easier with smaller teams and codebases.

In general: DDD alone will not be able to give you small enough microservices and/or solve data embedding requirements.

Data Storage Alternative:

Event Sourcing & CQRS

CRUD-Oriented System Model



"Event Sourcing is all about storing ***facts*** and any time you have "state" (structural models) – they are first-level derivative off of your facts.

And they are ***transient.***"

- Greg Young

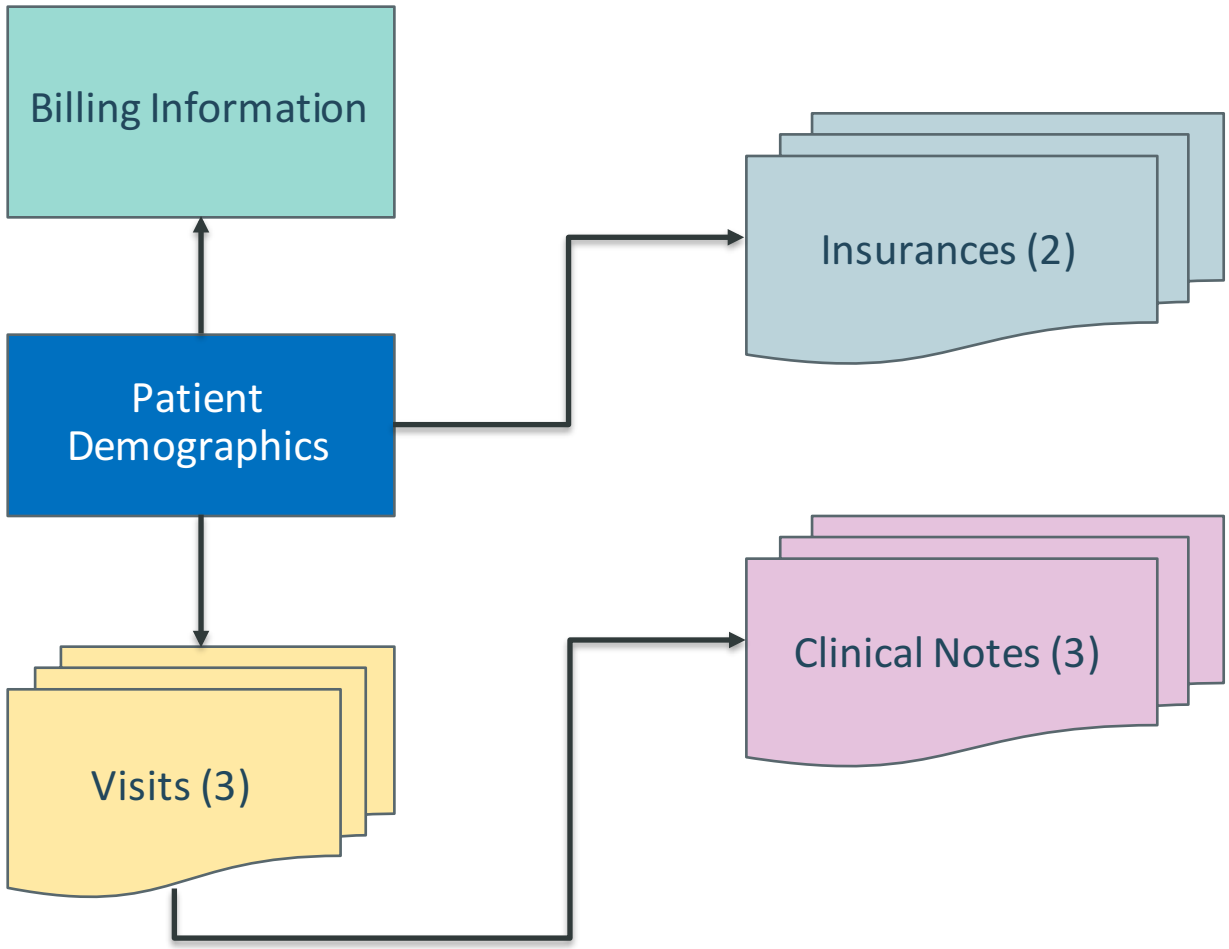
Command and Query Responsibility Segregation (CQRS)

Segregate operations that read data from operations that update data by using separate interfaces.

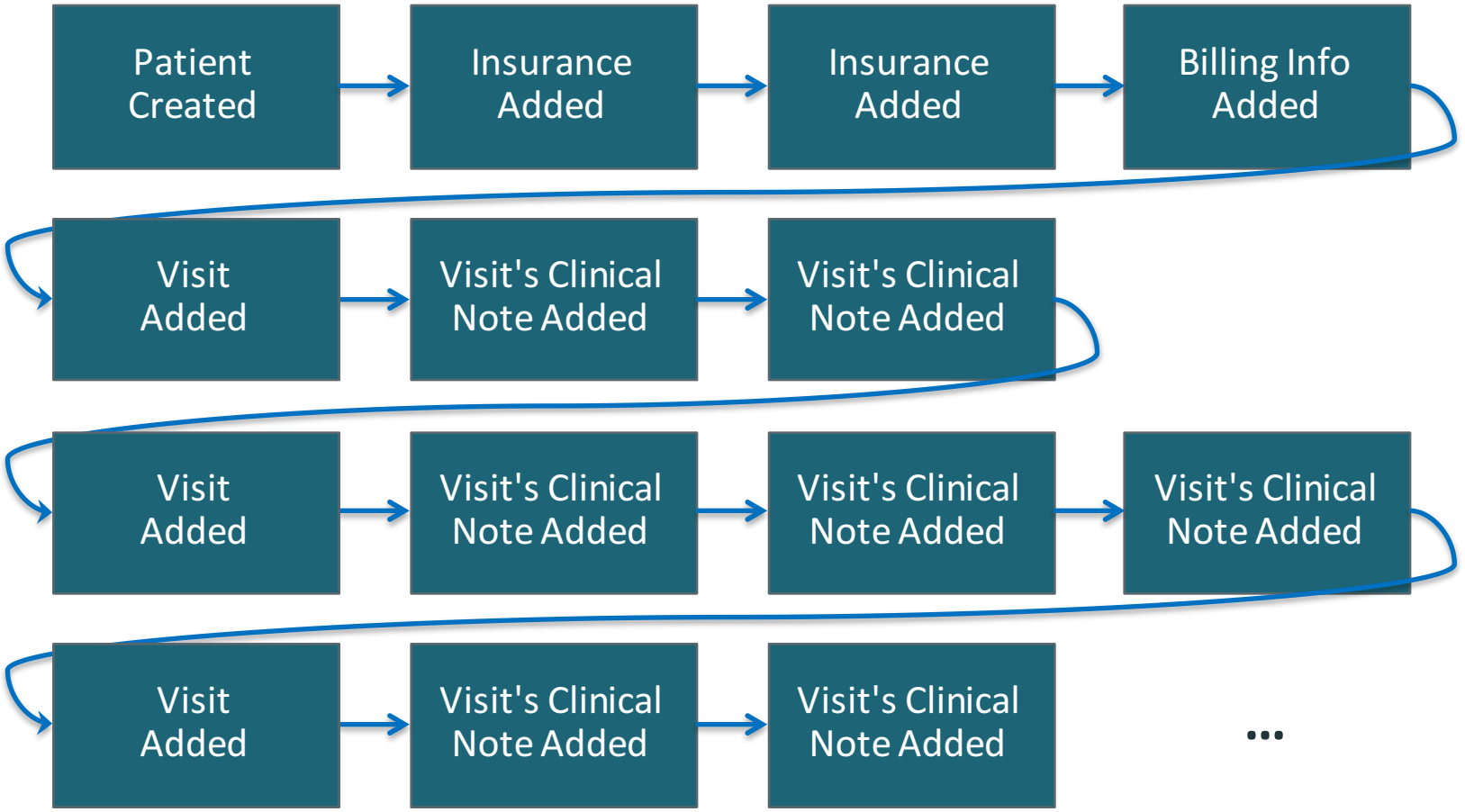
This pattern can maximize performance, scalability, and security; support evolution of the system over time through higher flexibility; and prevent update commands from causing merge conflicts at the domain level.

https://msdn.microsoft.com/en-us/library/dn568103.aspx

"Current State"-Oriented System Model

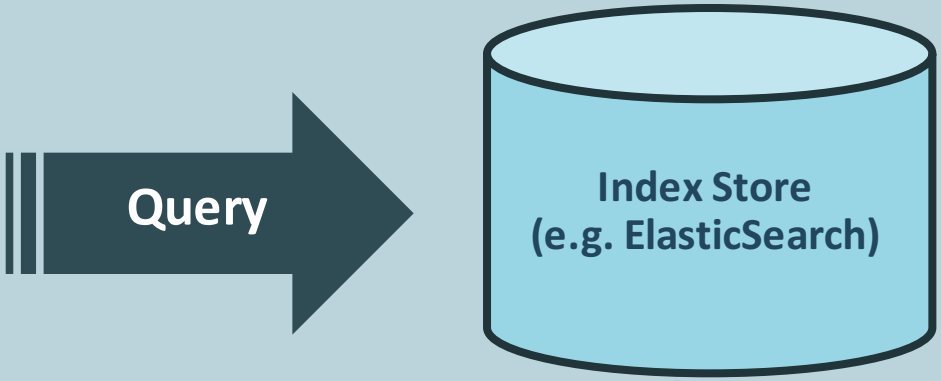


Events-based Model



"Patient Visits" Microservice

"List Patient Visits by a Doctor" Microservice



Do Not Overuse ES or CQRS!

You should only use Event Sourcing and CQRS when necessary. It is not an architecture for your entire system, but a tool to be used sparingly.

What About Them Transactions?

Use: Sagas

(Long-Lived Distributed Transactions)

Designed by: Hector Garcia-Molina & Kenneth Salem, Princeton, 1987

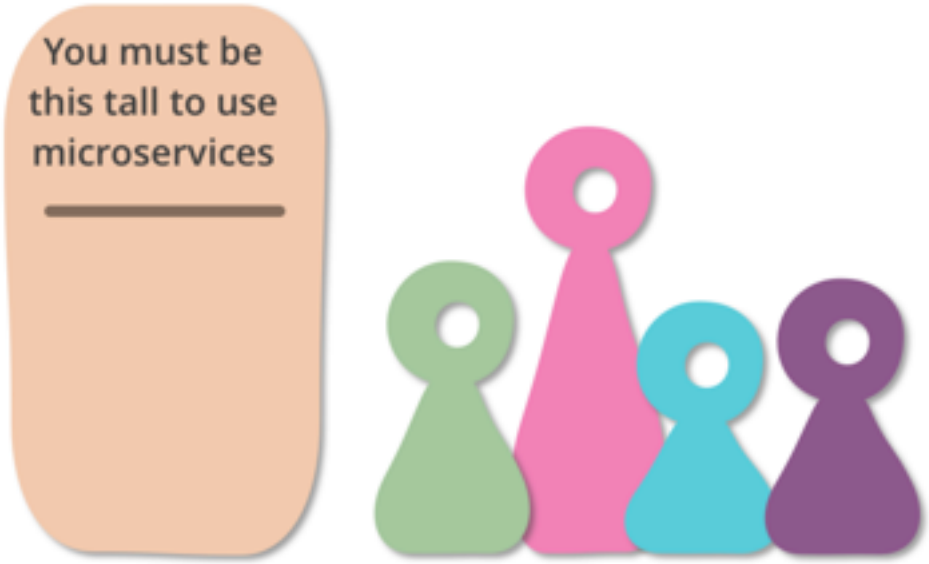
@see: *http://vasters.com/clemensv/2012/09/01/Sagas.aspx*

One More Thing...

MicroservicePrerequisites

Martin Fowler
28 August 2014

As I talk to people about using a microservices architectural style I hear a lot of optimism. Developers enjoy working with smaller units and have expectations of better modularity than with monoliths. But as with any architectural decision there are trade-offs. In particular with microservices there are serious consequences for operations, who now have to handle an ecosystem of small services rather than a single, well-defined monolith. Consequently if you don't have certain baseline competencies, you shouldn't consider using the microservice style.



I really, really hated this message, so I did smth. about it

http://martinfowler.com/bliki/MicroservicePrerequisites.html

https://github.com/apiacademy/microservices-deployment

Irakli Nadareishvili
Director of Strategy, API Academy

@inadarei
@apiacademy
@cainc

http://bit.ly/microservices-blindspots