

FOLLOW THE YELLOW BRICK ROAD

Taking Microservices Home in Large Enterprises

Zhamak Dehghani
@zhamakd
zdehghan@thoughtworks.com

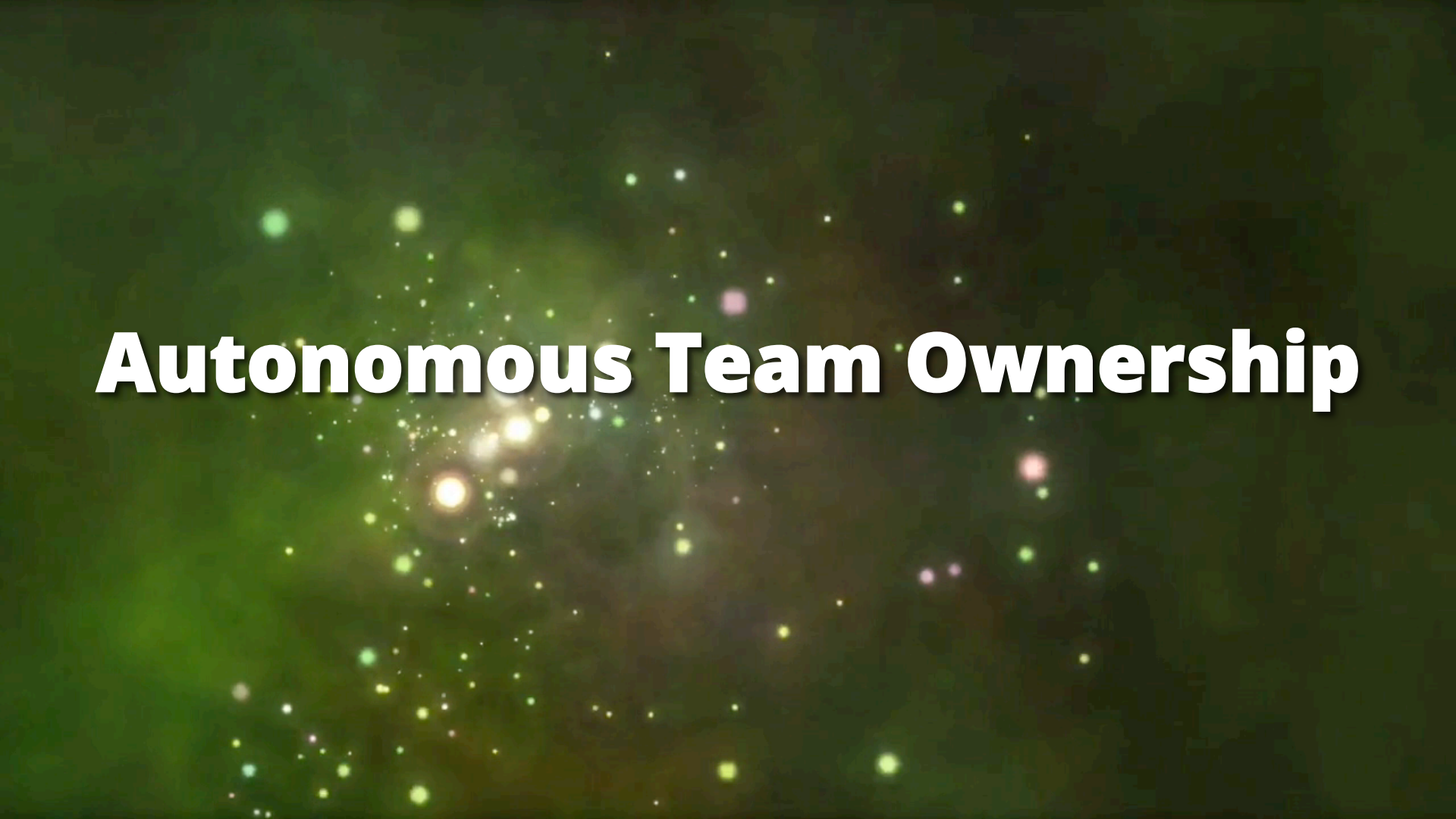


4 Years into the Microservices Architecture ...

Single Purpose Services



Built Around Business Domain



Autonomous Team Ownership



Collaborate via Choreography



Federated & Automated Delivery Pipelines



The Utopian World of Microservices

Follow the Yellow Brick Road with Me ...

Organizational Structure & Budgeting

Battling the Legacy Monoliths

Service Modeling

Operational Readiness



**IT IS NOT A TECHNICAL
PROBLEM**

Annual Portfolio Planning

Hard to sell re-architecture business case

Ephemeral project teams

No long term ownership

WINNING THE PORTFOLIO BUDGETING

- 
- The background image shows two men in traditional Chinese clothing performing a stunt in the air. One man, on the left, is in a dark robe and is holding a sword. The other man, on the right, is in a light-colored shirt and dark pants, and is holding a long pole. They are both in mid-air, with their bodies angled towards each other. Below them is the roof of a traditional Chinese building with dark tiles and ornate decorations. The sky is cloudy.
- Create a business incentive
 - Align business and tech initiatives
 - Service platform thinking vs. Feature thinking

FROM PROJECT TO PRODUCT TEAMS



NOT

~~THE~~

A - TEAM

PATH TO MICROSERVICES ORGANIZATION

- **Start with a Hypothesis**
- **Verify with a PoC**
- **Build a small cross-skilled team**
- **Expand and evolve to the rest of org**



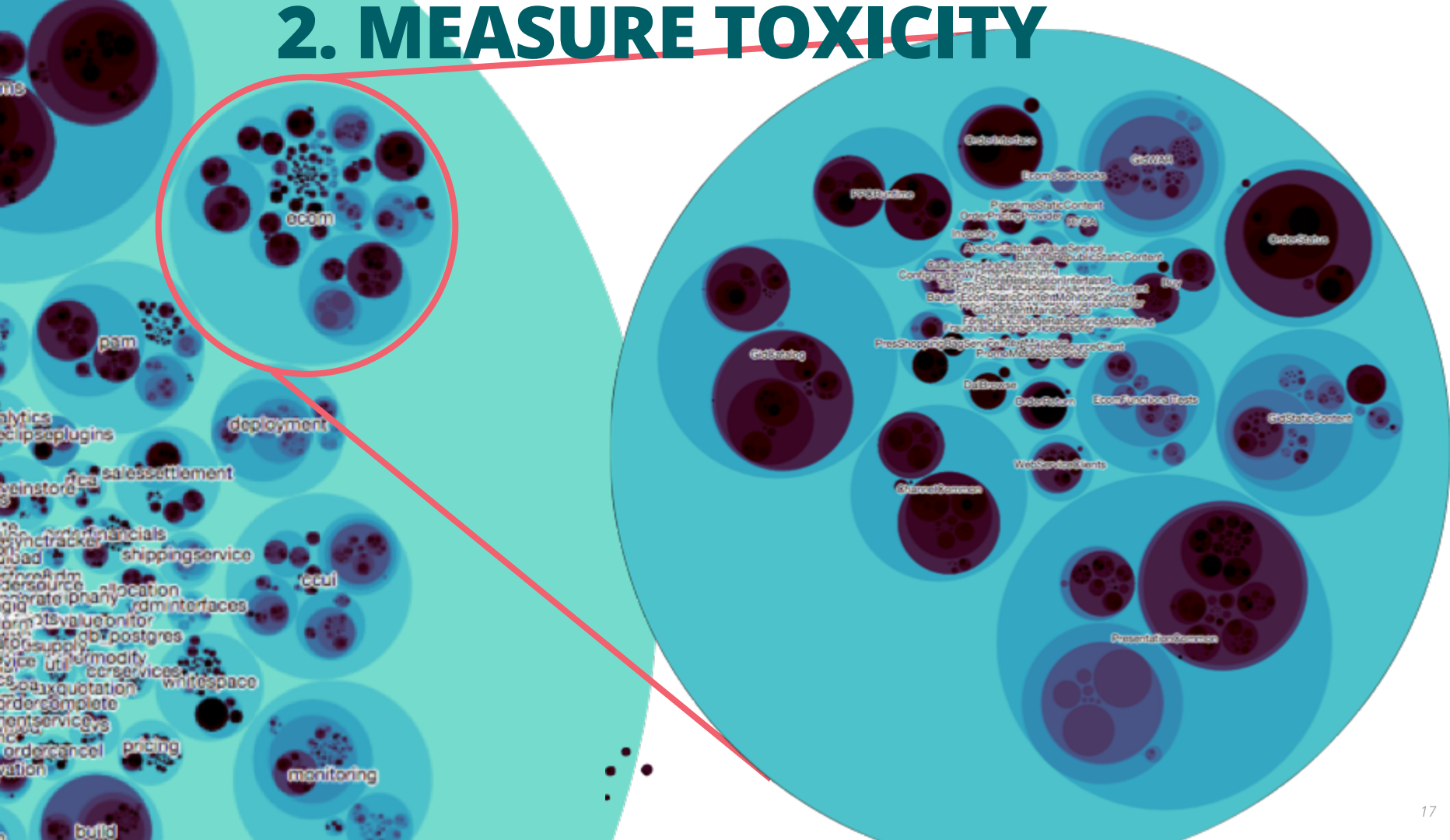


DECONSTRUCTING THE MONOLITH

Reuse & Extract vs. Rewrite/Buy & Retire

[illegible]

2. MEASURE TOXICITY



3. USE THEORY OF CONSTRAINTS TO FIND THE NEXT BOTTLE-NECK & EXTRACT



User A



User B



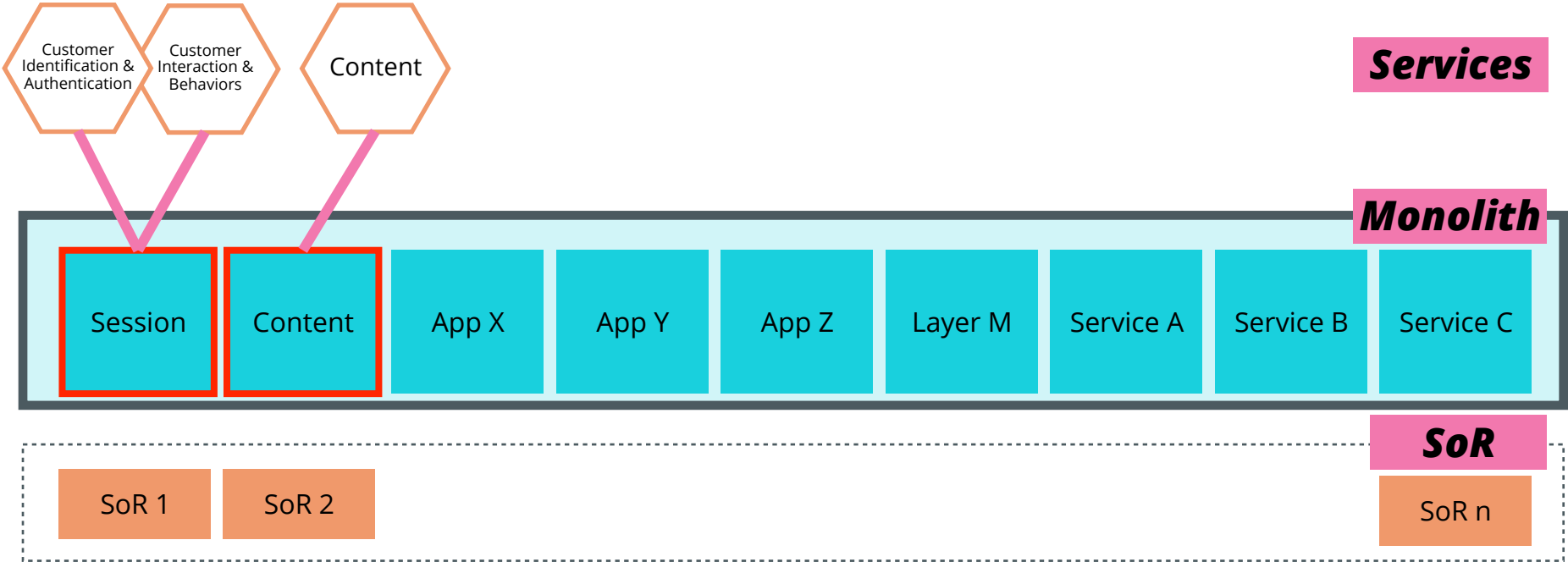
User C

Front end

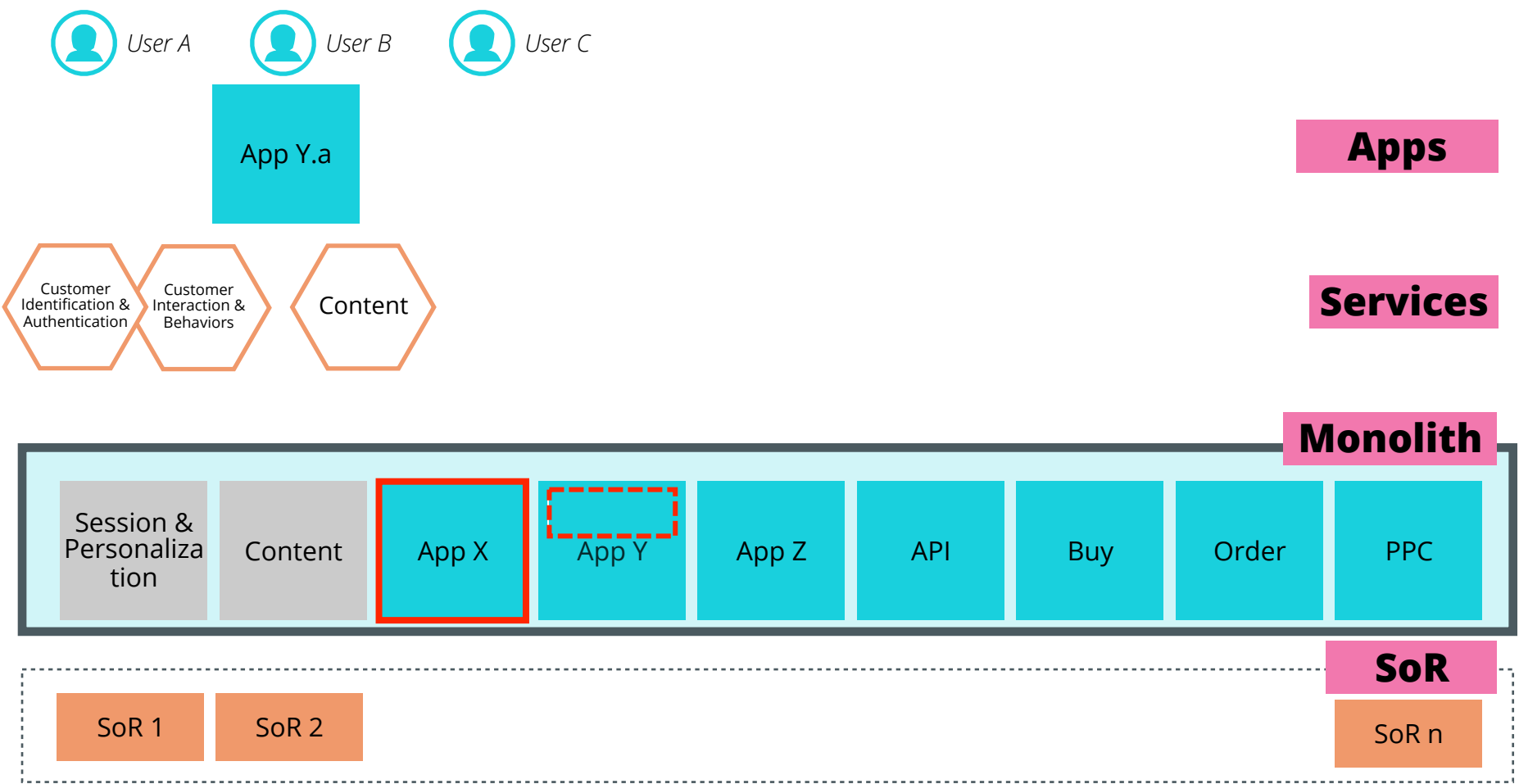
Services

Monolith

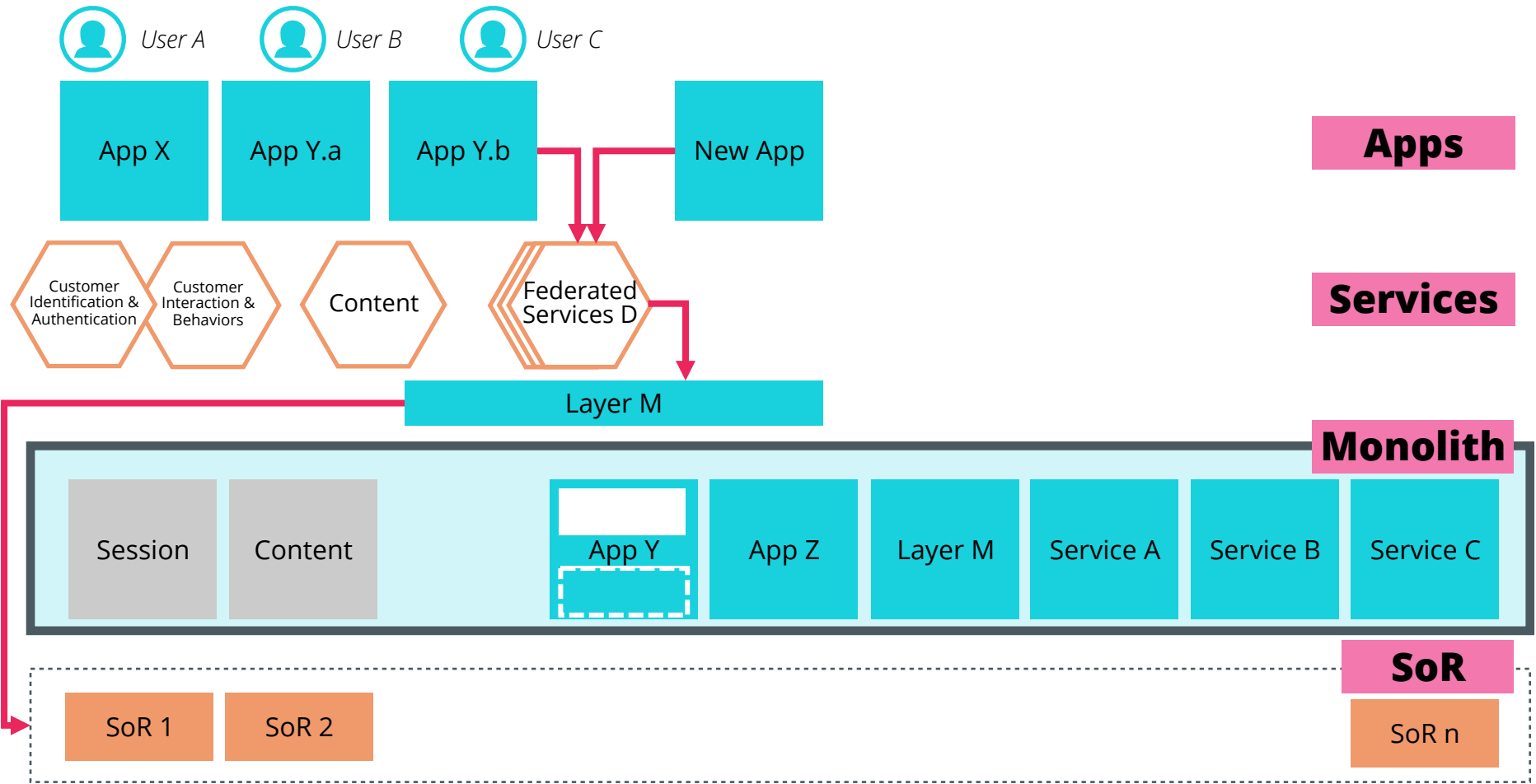
SoR



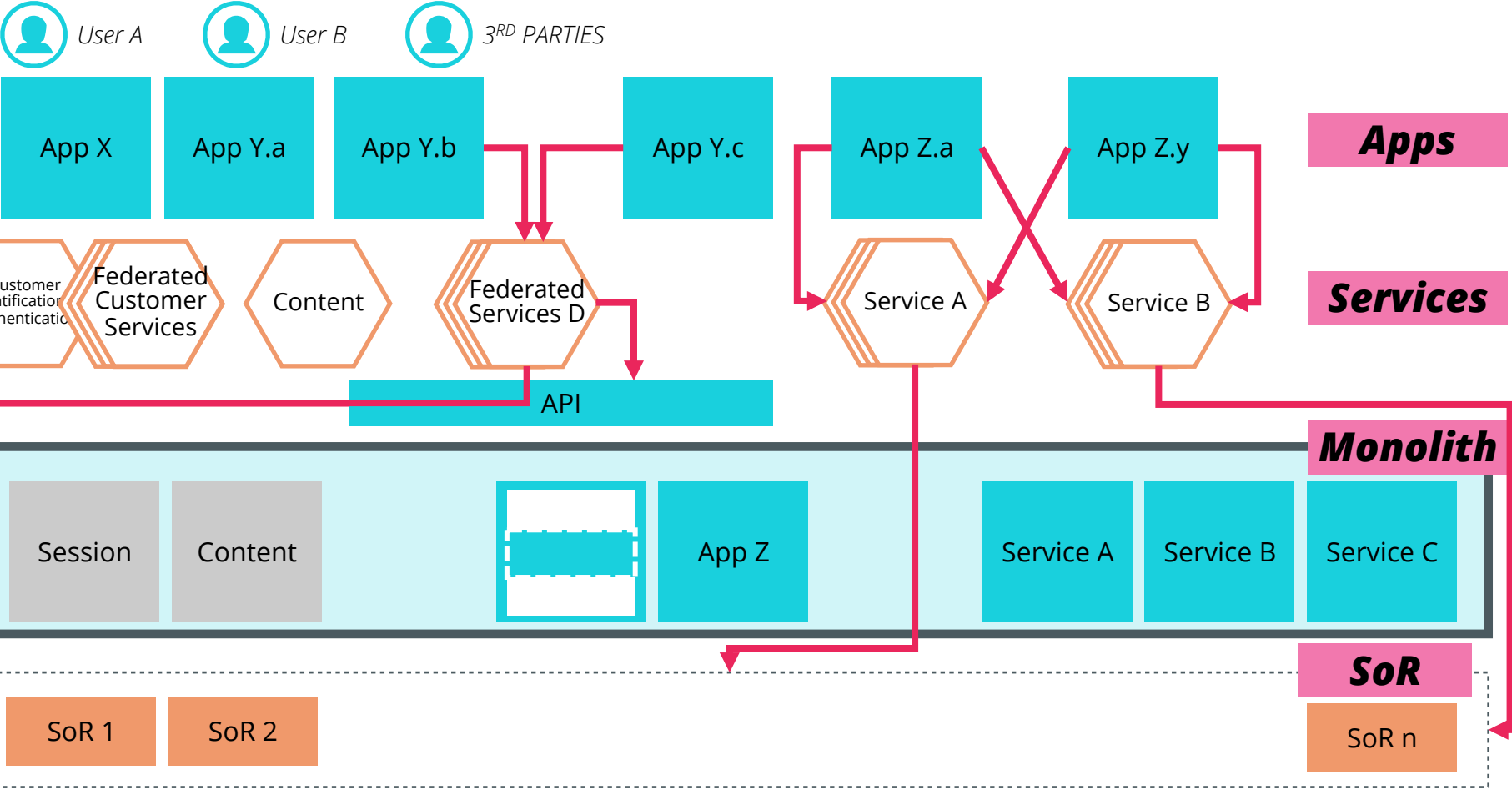
3. USE THEORY OF CONSTRAINTS TO FIND THE NEXT BOTTLE-NECK & EXTRACT



3. USE THEORY OF CONSTRAINTS TO FIND THE NEXT BOTTLE-NECK & EXTRACT



3. USE THEORY OF CONSTRAINTS TO FIND THE NEXT BOTTLE-NECK & EXTRACT



4. MONOLITH IN THE BOX



4. MONOLITH IN THE BOX

- Build and run monolith in a container
- First Principle:
 - Keep changes to monolith minimal
- Second Principle:
 - In a Hybrid setup any intermediate state

1 Find the seams around domain bounded context

2 Measure toxicity before reuse

3 Use theory of constraints to unlock decomposition

4 Apply strangler pattern

5 Keep the monolith in the box

6 Refactor what matters!



SERVICE MODELLING IS DIFFICULT

Curb the enthusiasm on going too micro too fast



**1. NOT TOO BIG NOT TOO SMALL
JUST RIGHT!**



1. NOT TOO BIG NOT TOO SMALL JUST RIGHT!

It depends ... on org maturity

Go macro then micro

CRUD services with no logic are too micro

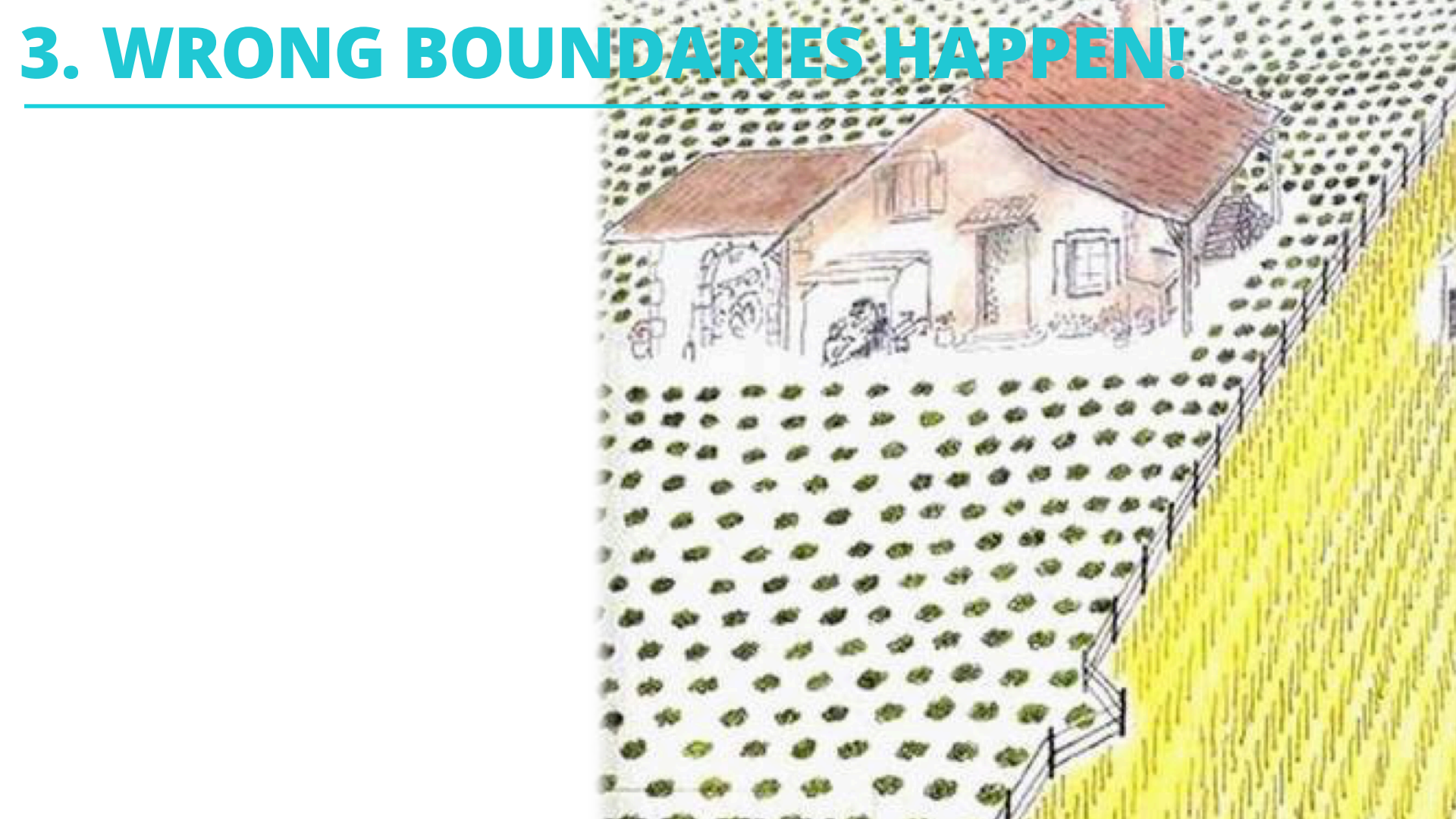
2. NO UTILITY SERVICES PLEASE

No config, utility, etc. shared services

Keep reference data with its entity domains

Simply duplicate!





3. WRONG BOUNDARIES HAPPEN!

3. WRONG BOUNDARIES HAPPEN!

**Use journeys to design services
Instead of round table discussions**

A confused consumer is a bad sign

**More than 3 services changed for a feature
is also a bad sign**

HATEOAS L3 Links to the rescue!



PRODUCTION AUTOMATION FIRST

No Service without an automated path to release

1

Contract tests first, then service implementation

In fact Consumer contracts first ...

2

Automate, automate, automate!

All the way to the production

3

Get the trade-offs on the infrastructure correct

*Correlation IDs for debugging, metrics for monitoring, etc.
implemented as independent shared libraries*

VALUES

Autonomy

Speed of Change

Scale

Composability

Tech Diversity

COMPLEXITIES

Communication

Execution

Resilience

Maintenance

Operational



MICRO



MACRO

EMPATHY

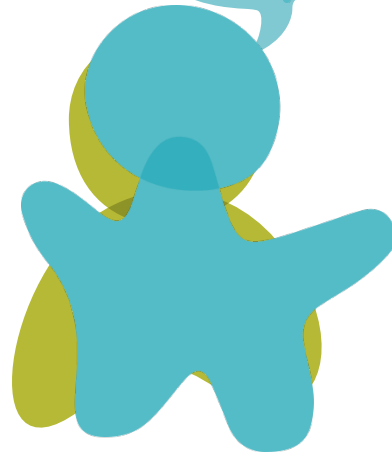


API First

*Consumer
Driven
Contracts*

*Service Provider
Mocks*

...



THANK YOU!

Zhamak Dehghani

zdehghan@thoughtworks.com

[@zhamakd](#)



Please

**Remember to
rate this session**

Thank you!

