Emergent Distributed Architectures:
Microservices and Data Pipelines

—

Sean Braithwaite @brapse · Peter Bourgon @peterbourgon · Velocity London 2017
Architectural decisions are expensive.
We want to answer not only how, but why and when.
Organization as organism.
Cells as parallel actors.
Teams as cells.
Teams enable parallelism.

Teams must consider global interests to avoid harmful local maxima.
Bounded contexts reduce external coördination costs & make internal work more efficient.
The microservices pattern.
We can be overeager applying apparently successful patterns.
Microservices cause more problems than they solve.
But microservices codify and reify their teams' responsibilities.
Well-defined teams and services maximize product velocity.
Microservices actually solve communication problems.
Microservices are an immune response to velocity gridlock.
The data pipelines pattern.
Data pipelines connect domains.
Variance in access patterns reveal problems and opportunities.
Domain experts should own ETLs.
Understanding resources for design.
Incidental coördination as failure.
Collaborative efforts rely on trust.
Domains need integrity to grow.
Good protocols maximize velocity.
Know when and how much to architect.
Organization as organism.
Microservices:
Why? — To solve velocity gridlock.
When? — It’s obviously necessary, and there are resources for automation.
Microservices:

How? — *Services with bounded contexts.*

Data pipelines:

**Why?** — To enable collaboration between domain experts.

**When?** — Domain expertise varies, and coördination is expensive.
Data pipelines:

**How?** — Explicit dependency structures.

**How?** — Autonomous, “vertical” teams.
Explicit boundaries maximize velocity.
Emergent Distributed Architectures:
Microservices and Data Pipelines

Sean Braithwaite @brapse · Peter Bourgon @peterbourgon · Velocity London 2017