Dreamforce

Design a DevOps Strategy for Multi-Org Implementations

Christopher Ramm

Sebastiano Schwarz

Forward-looking statements

This presentation contains forward-looking statements about, among other things, trend analyses and statements regarding future events, anticipated growth and industry prospects, and our strategies, expectation or plans regarding product releases and enhancements. The achievement or success of the matters covered by such forward-looking statements involves risks, uncertainties and assumptions. If any such risks or uncertainties materialize or if any of the assumptions prove incorrect, results or outcomes could differ materially from those expressed or implied by these forward-looking statements. The risks and uncertainties referred to above include those factors discussed in Salesforce's reports filed from time to time with the Securities and Exchange Commission, including, but not limited to our ability to meet the expectations of our customers; uncertainties regarding AI technologies and their integration into our product offerings; the effect of evolving domestic and foreign government regulations; regulatory developments and regulatory investigations involving us or affecting our industry; our ability to successfully introduce new services and product features, including related to AI and Agentforce; our ability to execute our business plans; the pace of change and innovation and our ability to compete in the markets in which we participate; and our ability to maintain and enhance our brands.

Coffee on us.

The first 4,000 attendees to provide feedback on this event will receive a $5 Starbucks gift card.*

1. Download the Salesforce Events mobile app.

2. Navigate to surveys.

3. Complete (4) session surveys and the Event Survey.

4. Redeem your gift at Badge Pickup in-person on the last day of the event.

*Restrictions apply. See terms and conditions at sforce.co/survey-terms


salesforce
Thank you

Who are we?

Christopher Ramm

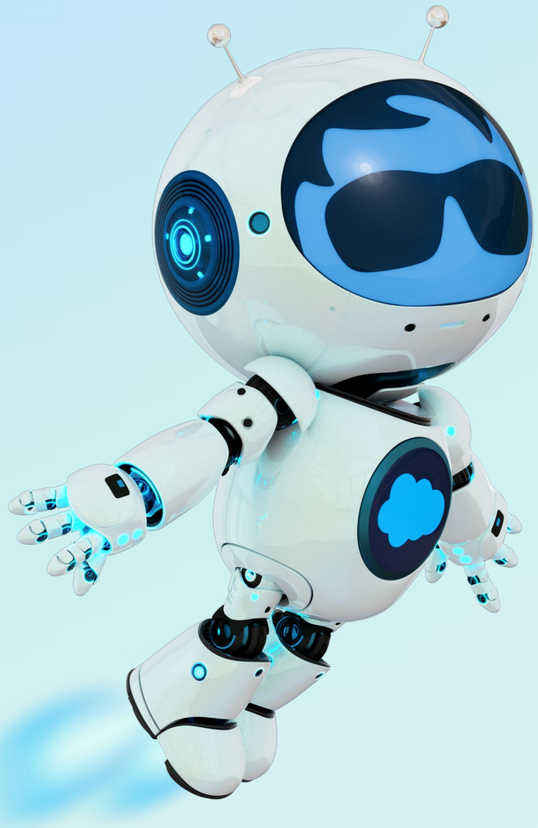
DCX CTO Germany & Salesforce CTA,
Capgemini

Sebastiano Schwarz

Salesforce CTO Germany,
Capgemini

Nobody ever simplified their Salesforce landscape by adding another org

Agenda



01 Why Multi-Org?

02 DevOps Approaches

03 Package-Based Delivery

04 Repository-Driven Delivery

05 Our Experience

06 Q&A

Why Multi-Org?

Five Dimensions of a Holistic Approach

Most pillars are in the organization's control

BUSINESS SETUP

- Strategy
- Business Unit setup
- Operation Model

GOVERNANCE AND CULTURE

- Sponsorship
- Business and IT Governance
- Vigilant

DATA STRATEGY

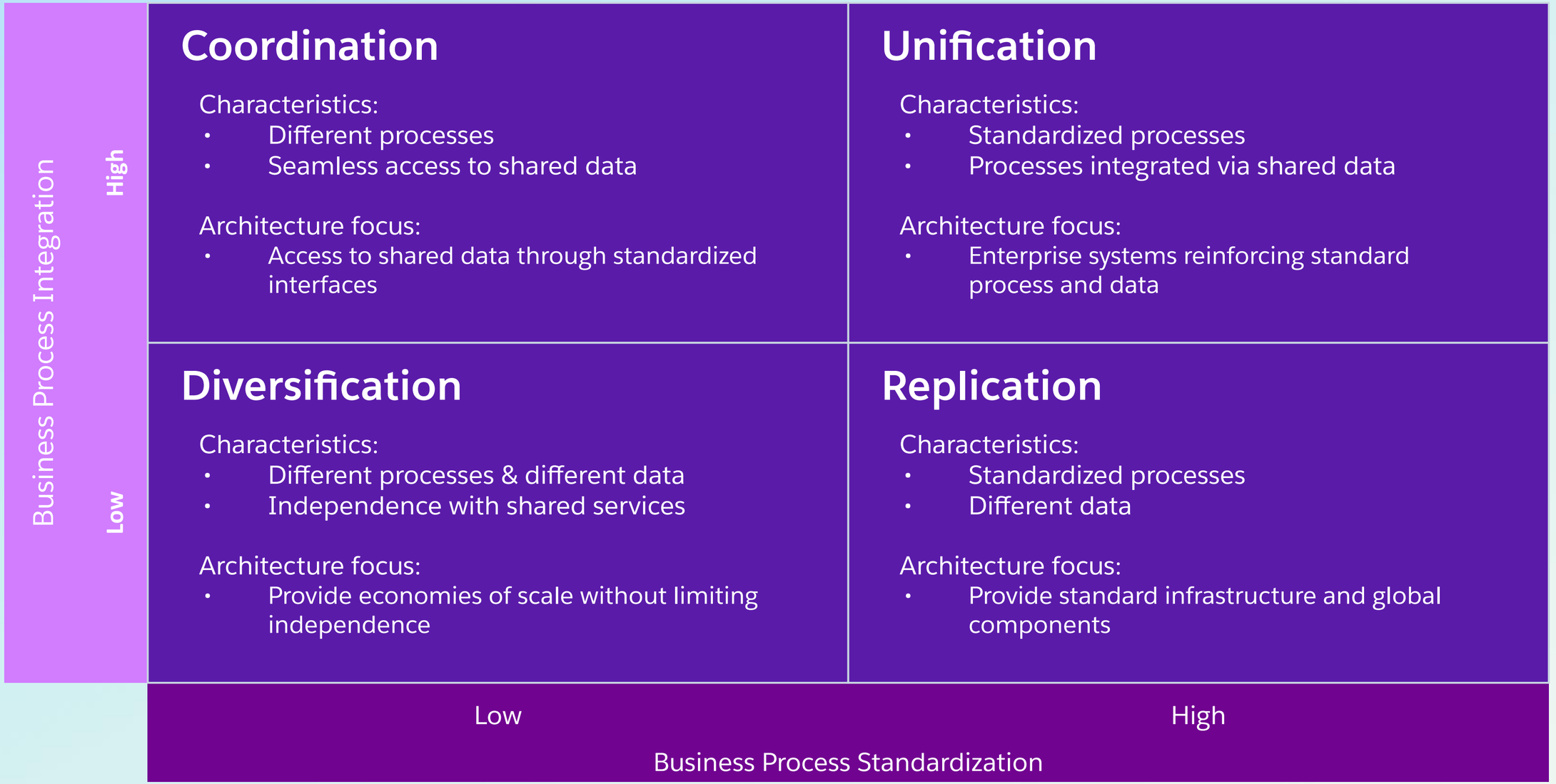
- Application architecture
- Functionality
- Data strategies
- Salesforce limitations

OPERATIONS (BUSINESS & IT)

- Design & delivery approach
- Deployment + end-user support
- DevOps

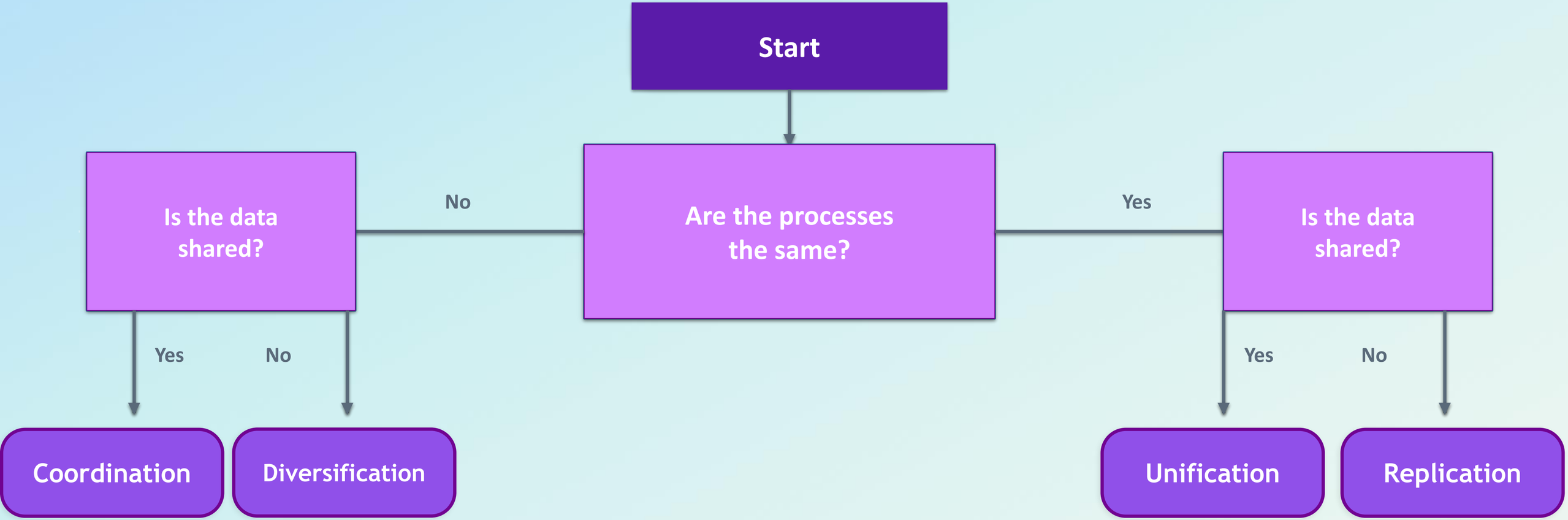
LEGAL REQUIREMENTS – MOSTLY OUT OF CONTROL

SOGAF Operating Models

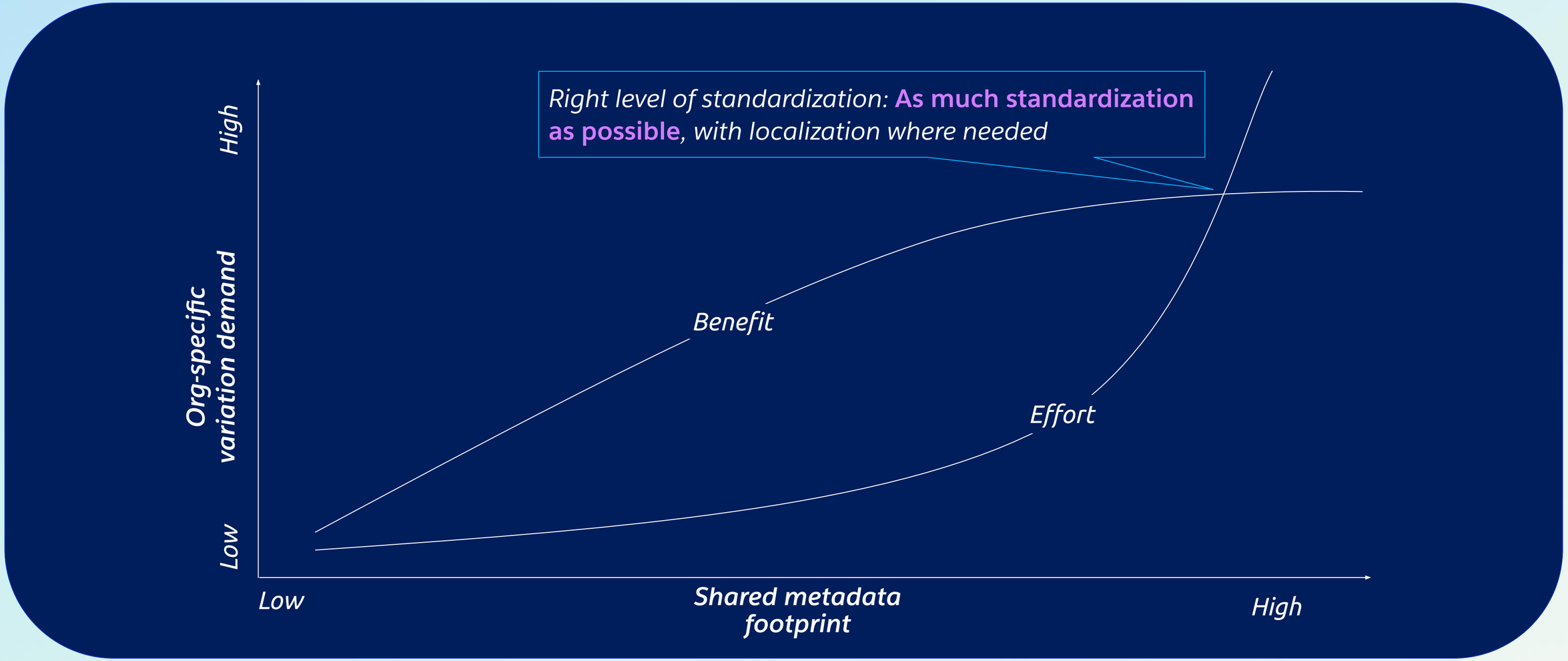


Based on MIT-CISR EA Model · Ross, Weill, Robertson · Adapted by Martin Griffiths (SOGAF)

Operating Model Decision Tree



Specific Requirements and Legacy Systems Influence Degree of Standardization



DevOps Approaches

Common Challenges in Salesforce DevOps

Cross-functional team setups

Modern DevOps practices require technical maturity that not all team members possess.

CAPABILITY GAPS

Lack of distributed responsibility

Shifting the entire DevOps responsibility to 1–2 people is a common antipattern, still causing bottlenecks in projects.

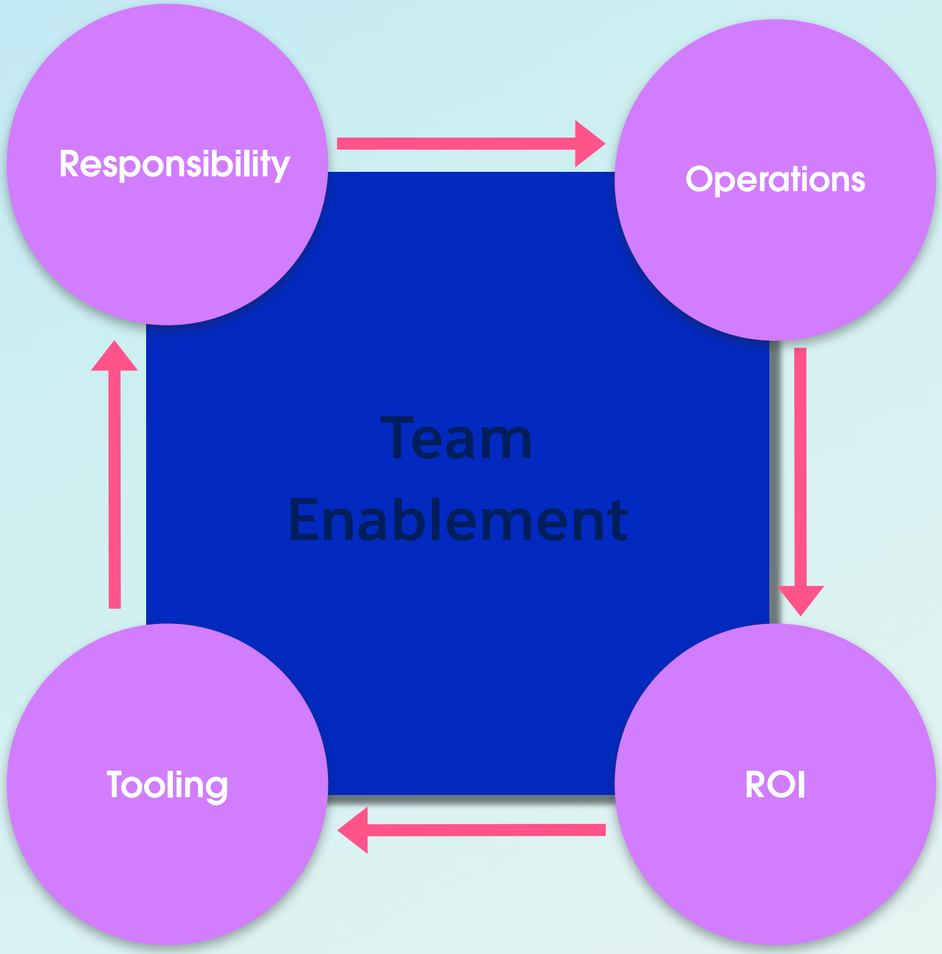
BOTTLENECKS

Wrong strategy for given project setup

Delivery approach, org strategy and team size are all factors to be considered for choosing the right strategy.

MISALIGNMENT

Build a DevOps Culture That Fits Your Needs



A strong DevOps culture depends on team enablement

There Are Two Common Approaches

Choose wisely!

Package-Based Delivery

- Components versioned & distributed as unlocked/managed packages
- Each org installs a defined package version independently
- Package registry = single source of truth

Repository-Driven Delivery

- Git repo holds canonical metadata, pipelines deploy to target orgs
- Changes tracked via branches, PRs, and merge policies
- Repo is source of truth – not the org
- Maximum flexibility, but demands strong branching discipline

Package-Based Delivery

Have a Set of Common Metadata

Why Packages

The deployment unit that makes multi-org manageable

Versioned – every release is a defined snapshot; roll back to any prior version

Modular – decompose your org into logical boundaries
(core, sales, service, integrations)

Dependency-aware – sfdx-project.json declares what depends on what

Repeatable – same package version installs identically across 2 orgs or 20

Independently upgradeable – Org A can run v3.2 while Org B stays on v3.1

Governance-friendly – package versions create an audit trail

Complexity Management

Three decisions you must get right before

01

Which Packages Are Org-Specific?

Separate what's shared from what's local – shared packages enforce consistency, org-specific packages preserve autonomy.

02

Dependency Management

Map your package dependency tree before you build your pipeline, not after your first cross-org deployment fails.

03

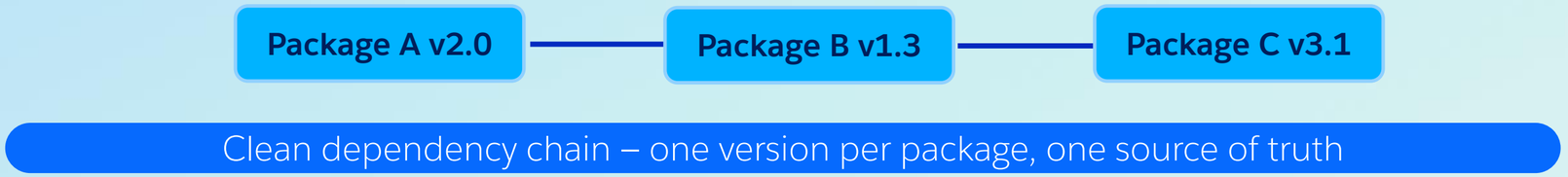
DevHub & Environment Strategy

Decide on one shared DevHub for visibility or per-BU DevHubs for autonomy

Why Packages

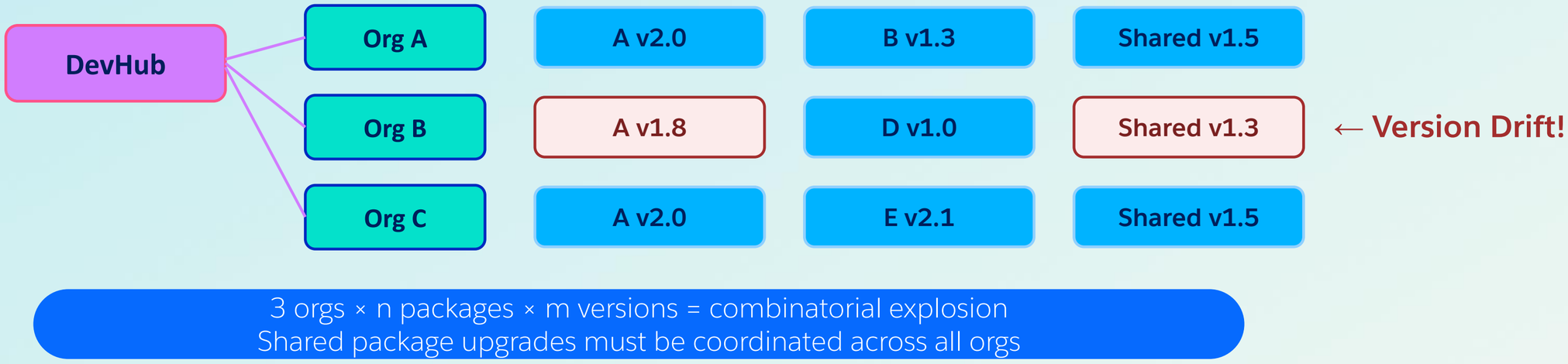
Single-org

One org, one DevHub, one pipeline

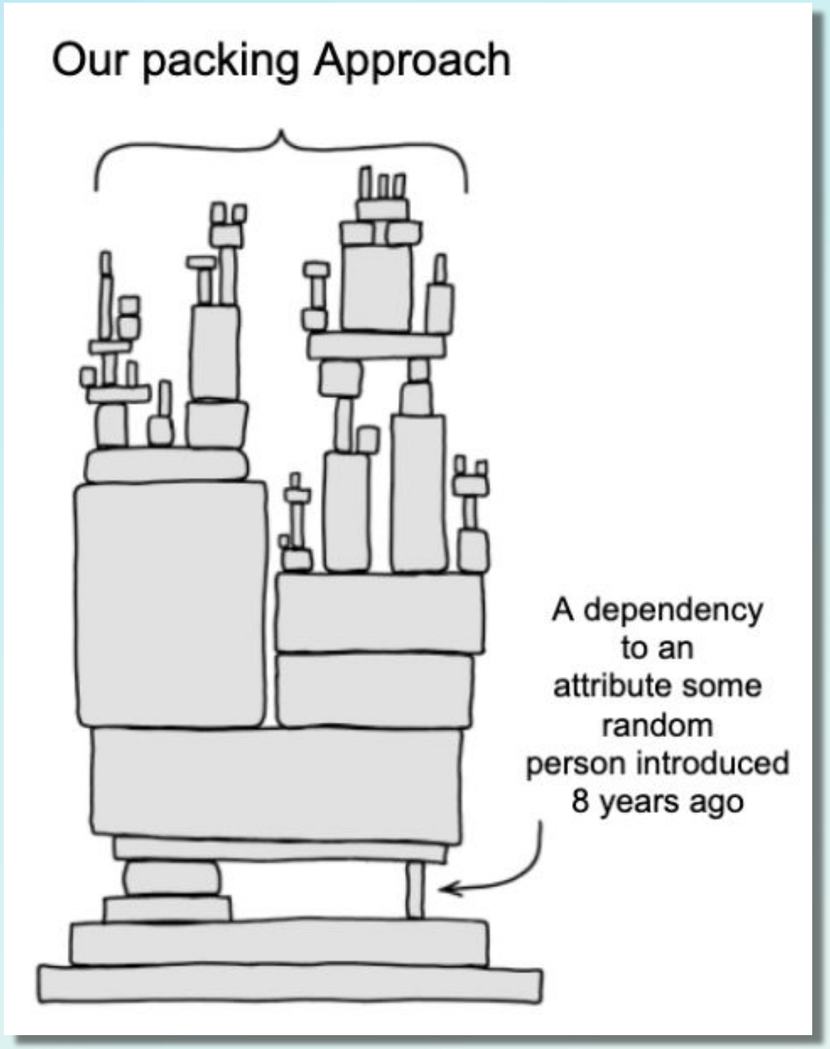


Multi-org

3 orgs, diverging versions, coordination overhead



We Learned It the Hard Way



Repository-Driven Delivery

Have a Set of Common Metadata

Why Repo?

Git as the single source of truth across all orgs.

Canonical state – the repo defines what each org should look like

Diff deployments – deploy only what changed between commits, not full metadata snapshots

Branch-per-org – org-specific overrides live in dedicated branches

Pull request governance – every change is reviewed and approved before it touches any org

Pipeline-as-code – deployment logic is versioned alongside metadata

Parallel promotion – promote the same commit to EU, US, APAC simultaneously

Complexity Management

Three decisions you must get right before

Monorepo or Multi-Repo?

One repo with directory-per-org gives central visibility and atomic cross-org commits

Multi-repo gives autonomy but sacrifices cross-org traceability

Branch & Merge Strategy

Define who merges what and when

Without clear merge policies, your repo becomes the new source of drift instead of the cure

Pipeline Orchestration

A single sf project deploy won't cut it. You need a sequenced fan-out: shared components first, org deltas second, smoke tests third – with per-org rollback capability built in, not bolted on

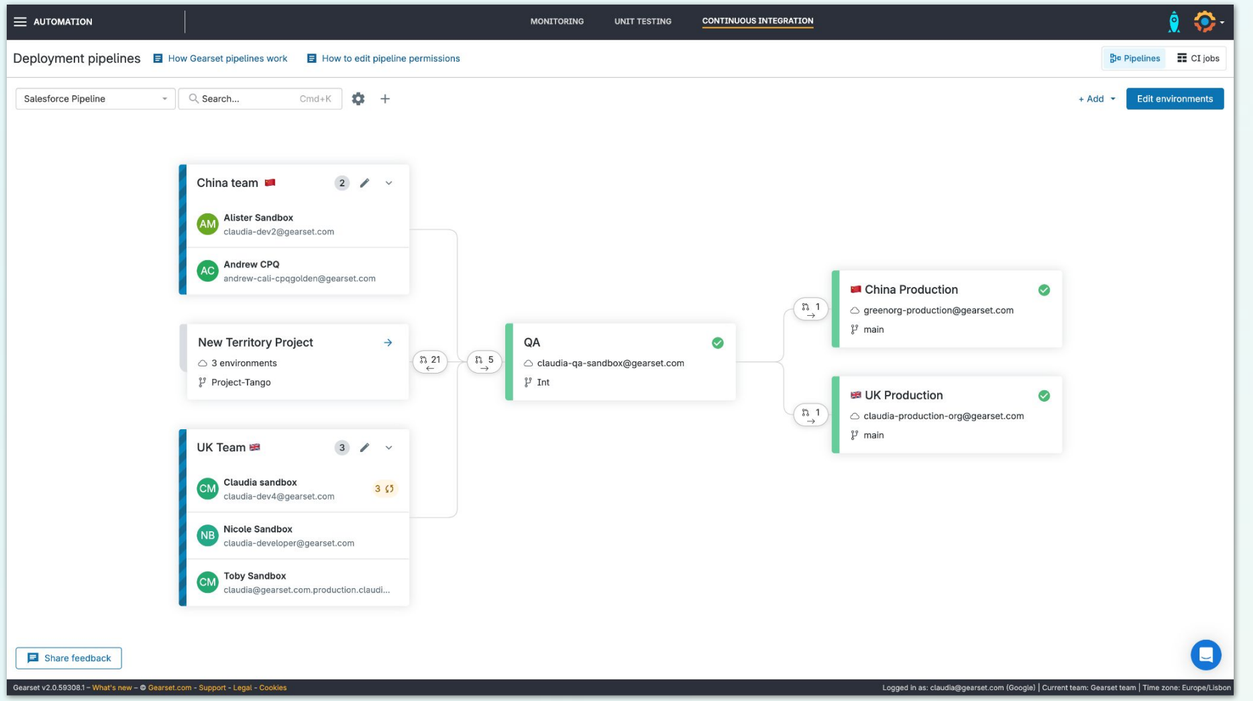
We Learned It the Hard Way

Our Experience

Multi-Org is not a Tooling-Issue. It's a Design Issue.

You don't have to build it all yourself

ISV partners offer DevOps tooling that accelerates adoption and some offer support for D360 and other benefits (e.g., testing, deployment steps)



ISVs provide a UI to handle your pipelines

Sneak

React & Angular in Multi-Org DevOps

Cross-functional team setups

Versioned like packages, shared UI upgradable per org

Lack of distributed responsibility

npm builds run alongside, not instead of, your metadata pipeline

Wrong strategy for given project setup

One UI can span multiple orgs via APIs, a new dependency to design for

If you remember nothing else from this session

1. **Start from the operating model, not the toolchain**
   Your org strategy is an enterprise architecture decision

2. **The pipe between orgs is harder than the pipe within each org**
   Single-org DevOps patterns break at the org boundary

3. **Make every architectural decision explicit — and document it**
   If these decisions live in someone's head instead of your repo, you don't have a strategy

4. **Design for Alforce, Agentforce and Data 360 today**
   Agent definitions and D360 tenancy introduce cross-org dependencies that will grow

GITHUB REPOSITORY

Dreamforce

Thank you