



SPA MDR CDR SCE

OS20: Hybrid SAS–R–Python Pipelines: Coordinating Multi-Language Workflows

Bill Qubeck and Ashwin Chaudhari



Agenda

- Introduction, Motivation, & Background
- Framework Principles
- Core Components of the Reference Architecture
- Case Study
- Future Outlook and Conclusion



Introduction



- Regulated analytics shifting to SAS plus R plus Python pipelines
- More capability, harder reproducibility, coordination, audit readiness
- **Main failure mode:** loss of control & lack of end-to-end traceability; not math errors
- Handoffs, parameters, schemas, dependency drift, missing evidence
- Rework and cycle time up, audit exposure, hard to recreate history
- **Orchestrated solution:** metadata coordination, package control, logging, RBAC, GxP ready



Challenges Integrating SAS, R, & Python Workflows



Framework Principles

Five core
five
principles

01 Shared Metadata

Define standardized input/output interfaces for SAS, R, and Python steps to ensure consistent communication.

02 Centralized Package MGT

Maintain validated versions of R and Python packages, and control SAS macros, enabling reproducible executions.

03 Harmonized Logging

Unified logging captures step execution, inputs/outputs environment details, and errors for audit readiness.

04 Role-Based Access Controls

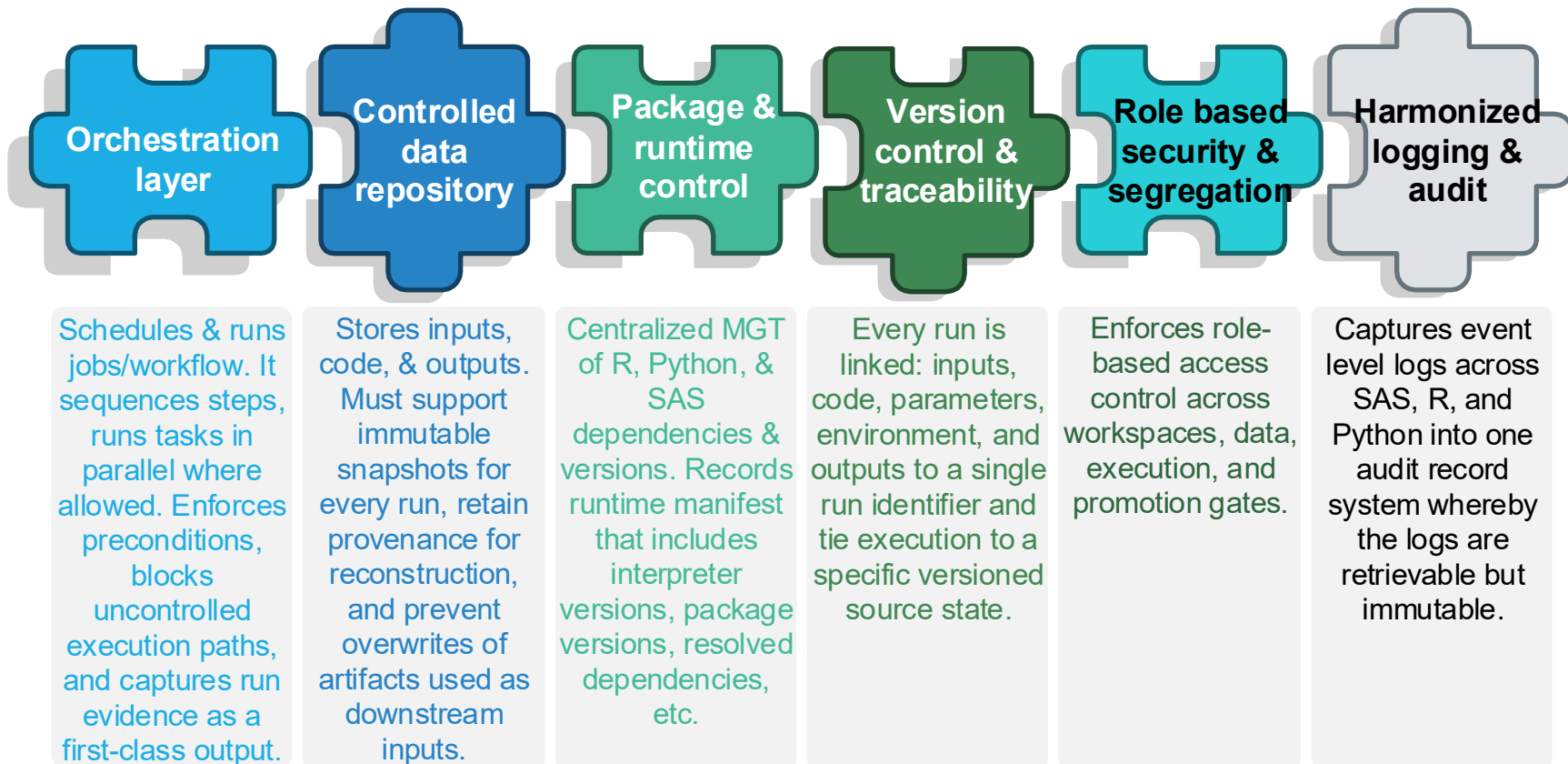
Protect workflow integrity by enforcing execution, modification, and access (e.g., read) permissions.

05 Flexible Orchestration

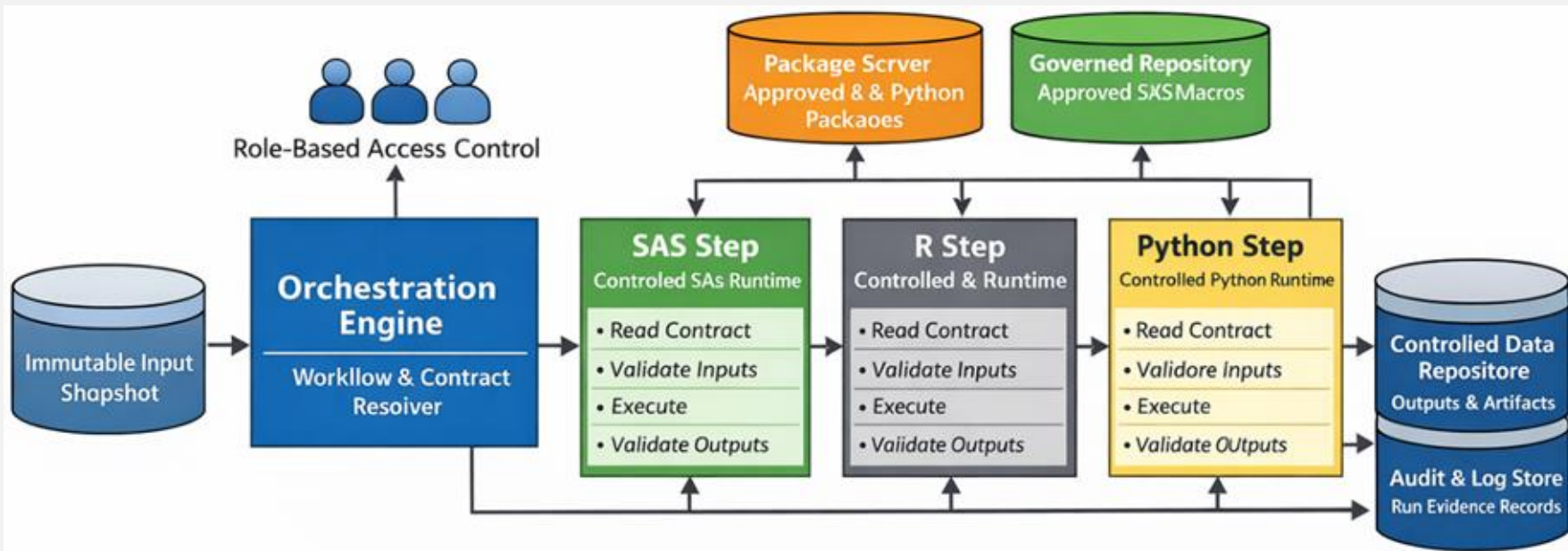
Support sequential, parallel, and conditional execution of heterogeneous steps.



Core Components of the Reference Architecture



SAS-R-Python Pipeline Architecture



Key Control Points for a Defensible Architecture

Contract enforcement at runtime

Inputs and outputs are checked and verified against the contract before downstream steps proceed.

01

Deterministic execution

Fixed seeds and pinned dependencies are required where results for regulatory reporting.

02

Promotion gates

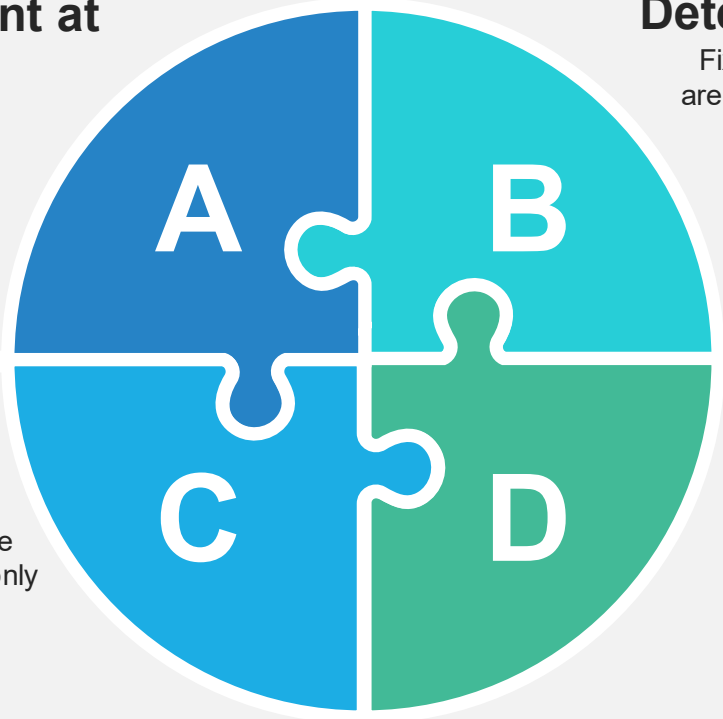
Contracts, code, and packages move from development to validated use only through documented review and approval.

03

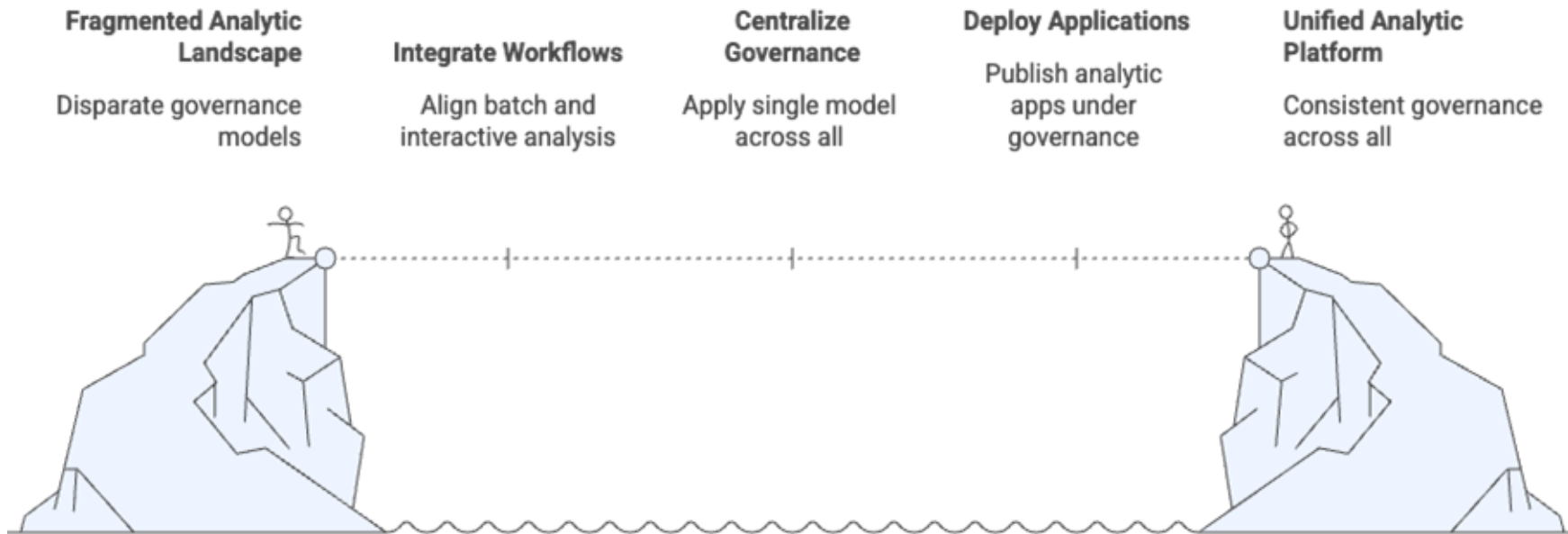
Evidence by design

Every run produces a complete evidence bundle without manual compilation.

04



Operating Model: One Governance Plane



Case Study: Integrated Pipeline

- **Data extraction and transformation (SAS):** Curated and standardized raw clinical data into analysis ready products aligned to the reporting cadence as well as for data cleaning activities.
- **Analysis and statistical methods (R):** Produced safety summaries and analytical outputs under an R first delivery model.
- **Visualization and publishing (Python):** Generated figures, dashboards, and fixed format deliverables required for review and regulated reporting packages.

Mid-sized pharmaceutical organization

Weekly clinical safety reporting

Python – Visualization & reporting

Visualization and reporting

Reduced manual handoffs

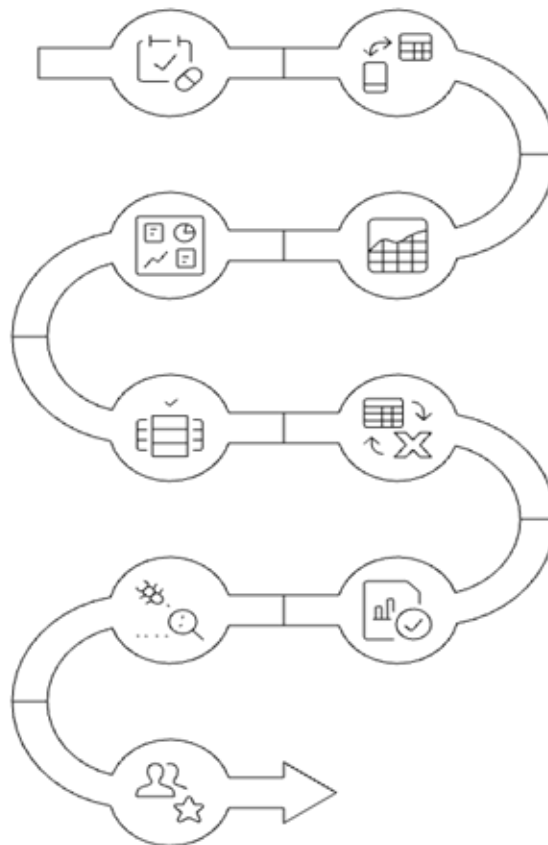
Streamlined workflow

Faster inspection response

Efficient response

Increased reviewer confidence

Enhanced trust



SAS – Data extraction & transformation

Data processing and preparation

R – Statistical modeling

Statistical analysis and modeling

Deterministic reproducibility

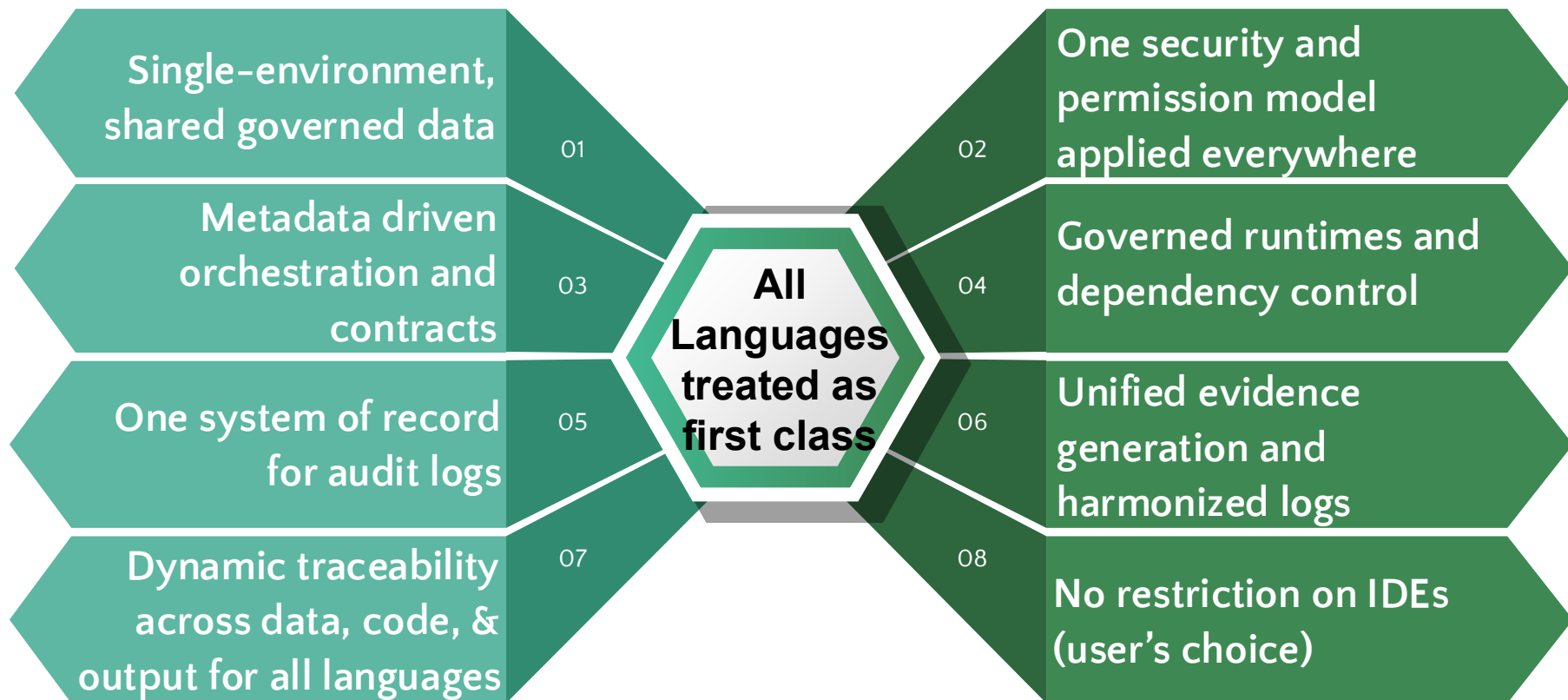
Consistent results

Audit-ready evidence bundles

Compliance and readiness



Implementation Foundation



Outcomes

01

Reduction in administrative overhead

Defining security and permissions once in the system and applying them everywhere removed duplicated administration across separate SAS, R, and Python systems. This reduces the ongoing cost of onboarding, role changes, and periodic access reviews.

02

Improved compliance through consistency not policy

A unified environment replaced procedural controls with enforceable controls. With one permission model, one governed data layer, one orchestrated execution path, one system of record for audit logs and evidence model, the organization reduced silent environment changes, incomplete audit trails, etc.

03

Faster root cause analysis and reproducibility

When discrepancies occurred, the run evidence showed exactly which data, code, output, configurations, and other artifacts were used across SAS, R, and Python.

04

R first strategy achieved

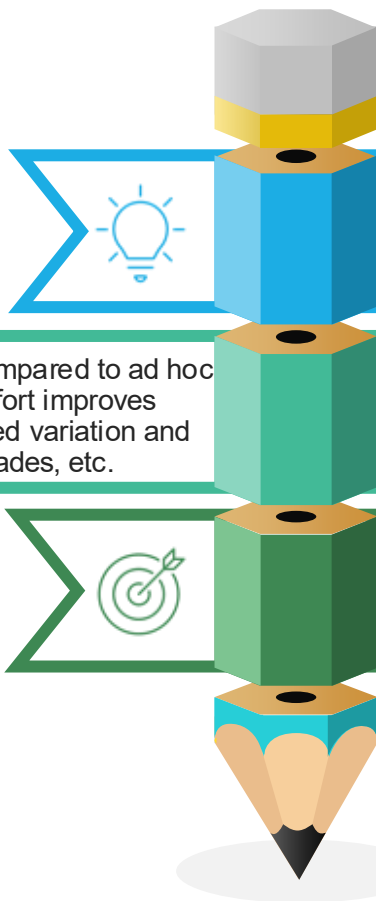
Proved R could be used as the primary analysis and reporting language without creating a parallel environment. Strategic goal of R first achieved - while still supporting SAS and did so without fragmenting governance, duplicating data, deploying additional tools, or splitting compliance responsibility across tool specific silos.



Three Key Lessons Learned

Foundational investment enabling long-term stability and scalability

The implementation reduces complexity compared to ad hoc &/or tool-specific workflows. This upfront effort improves long-term stability by preventing uncontrolled variation and minimizing rework during inspections, upgrades, etc.



Organizational maturity in governance, package management, etc.

Successful adoption requires a baseline level of organizational maturity, including defined ownership of packages, workflow governance, etc. It directly contributes to robustness by reducing ambiguity, enforcing consistency, and aligns with GxP expectations.



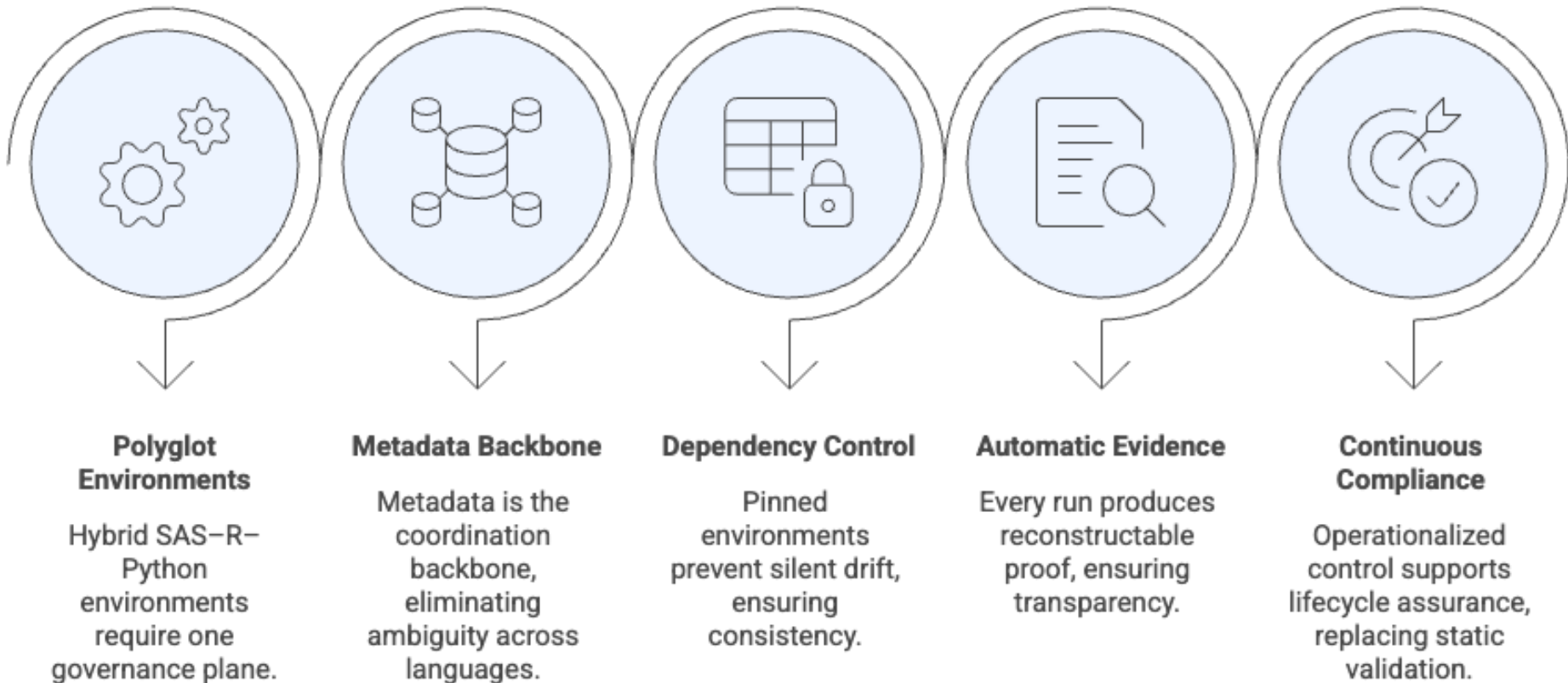
Approach reinforces robustness by ensuring that all workflow components adhere to defined interfaces and execution rules, which is essential for maintaining reproducibility, traceability, and audit readiness across heterogeneous analytics stacks.



Dependence on enforceable orchestration practices



Five Strategic Takeaways



Future Outlook and Conclusion:

Integrated SAS, R, and Python pipelines are inevitable and required!

- **Coordination as a platform capability:** Orchestration standardizes sequencing, branching, and evidence capture across languages.
- **Contracts make intent enforceable:** Shared metadata contracts drive consistent runtime behavior and reduce rework.
- **Inspection ready evidence by design:** Harmonized logging and run identifiers link configurations, inputs, outputs, and logs into reconstructable audit evidence.
- **Governed flexibility:** Centralized dependency governance, role-based access control, and controlled outputs let teams use the right tool without fragmenting compliance.



Thank You

<https://www.sycamoreinformatics.com>

See what's possible today: Booth #31



Bill Qubeck

Bill.Qubeck@SycamoreInformatics.com
<https://www.linkedin.com/in/williamqubeck/>



Ashwin Chaudhari

Ashwin.Chaudhari@SycamoreInformatics.com
<https://www.linkedin.com/in/ashwinchaudhari/>

