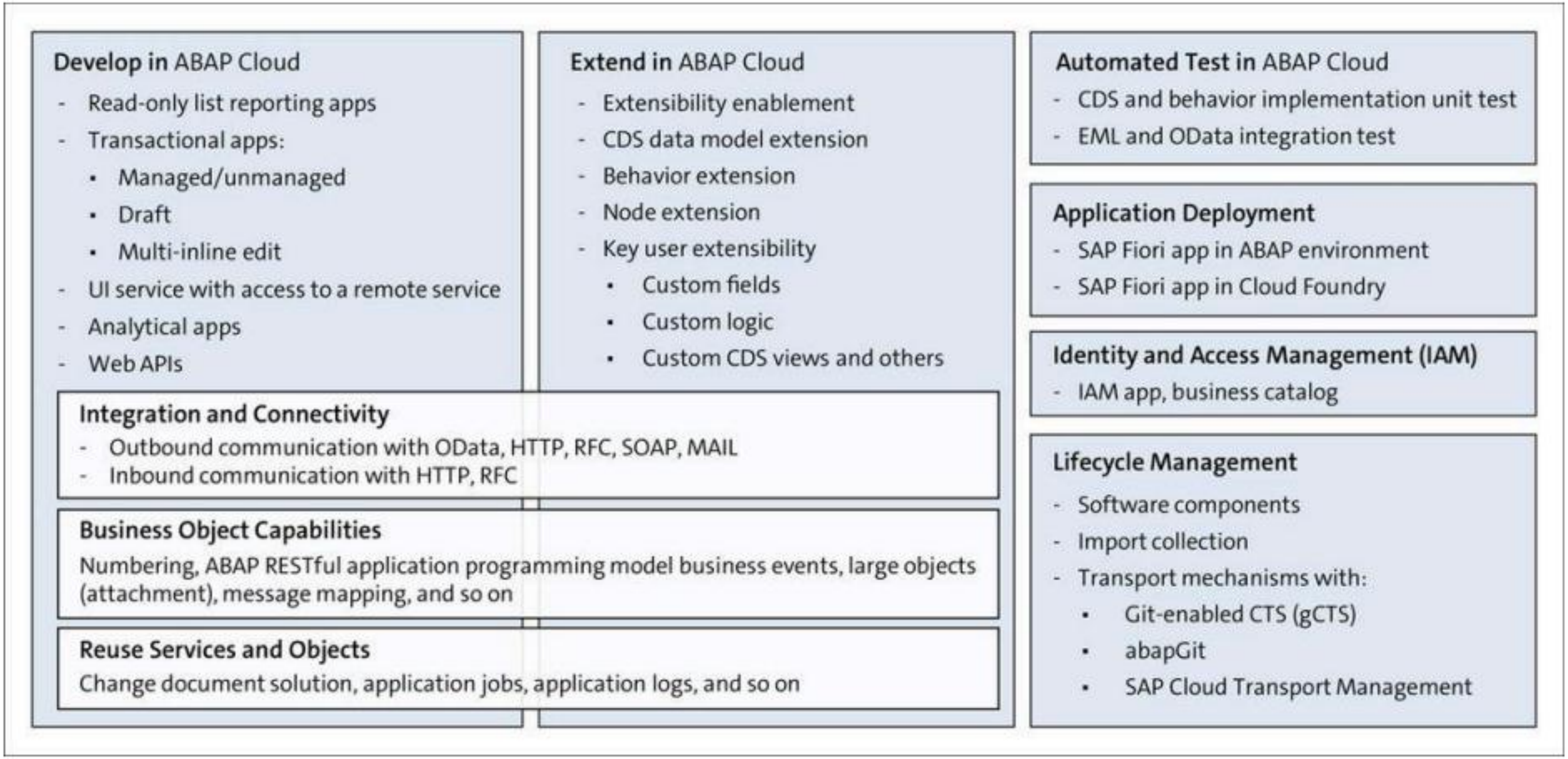
The ABAP Cloud Development Model for Modern Extensions

We'll focus on the ABAP Cloud development model, which forms the foundation of Tier 1 extensions. This model is critical for building future-proof applications in SAP environments.

por Mayko Silva



Develop in ABAP Cloud

- Read-only list reporting apps
- Transactional apps:
  - Managed/unmanaged
  - Draft
  - Multi-inline edit
- UI service with access to a remote service
- Analytical apps
- Web APIs

Extend in ABAP Cloud

- Extensibility enablement
- CDS data model extension
- Behavior extension
- Node extension
- Key user extensibility
  - Custom fields
  - Custom logic
  - Custom CDS views and others

Automated Test in ABAP Cloud

- CDS and behavior implementation unit test
- EML and OData integration test

Application Deployment

- SAP Fiori app in ABAP environment
- SAP Fiori app in Cloud Foundry

Identity and Access Management (IAM)

- IAM app, business catalog

Integration and Connectivity

- Outbound communication with OData, HTTP, RFC, SOAP, MAIL
- Inbound communication with HTTP, RFC

Business Object Capabilities

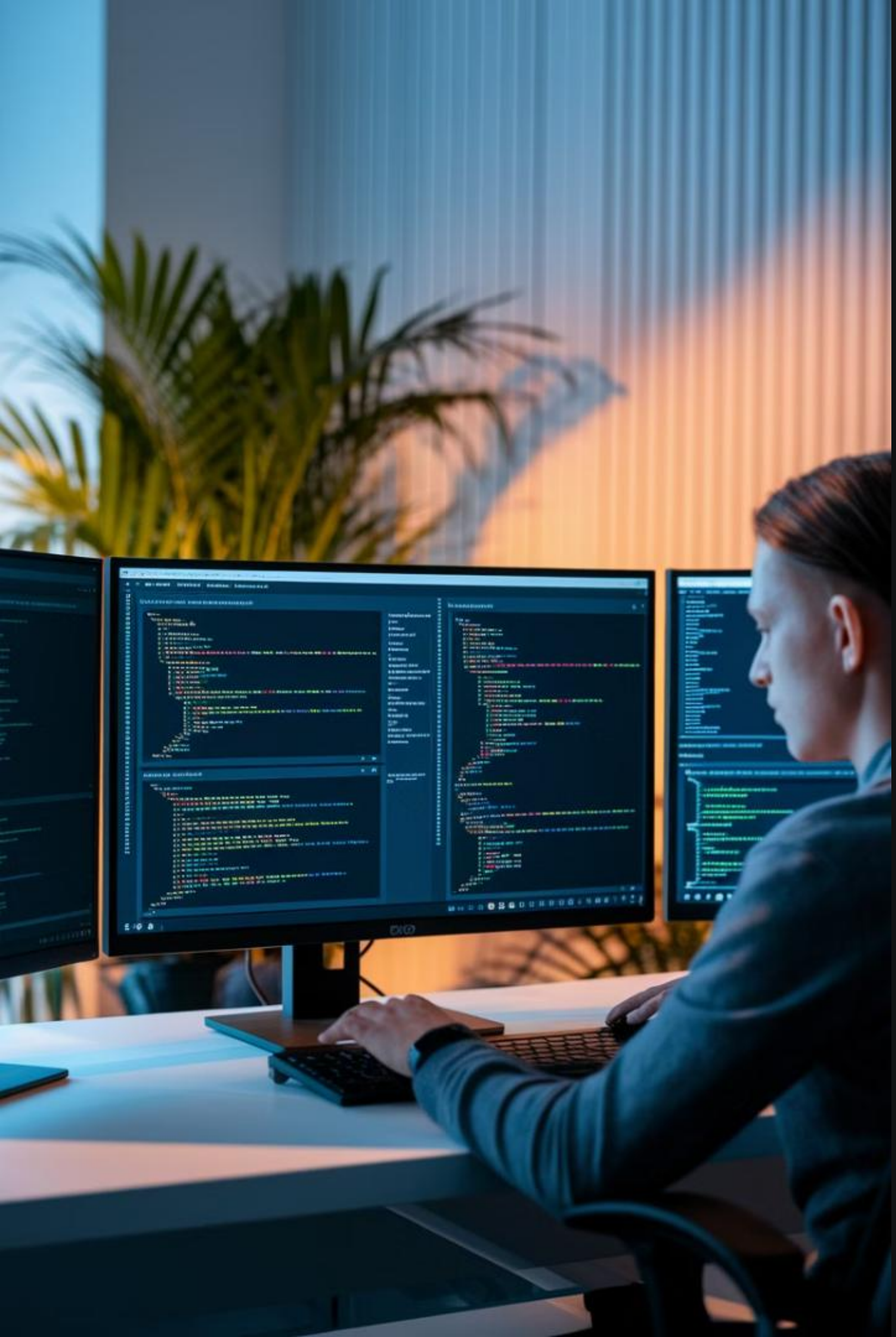
Numbering, ABAP RESTful application programming model business events, large objects (attachment), message mapping, and so on

Reuse Services and Objects

Change document solution, application jobs, application logs, and so on

Lifecycle Management

- Software components
- Import collection
- Transport mechanisms with:
  - Git-enabled CTS (gCTS)
  - abapGit
  - SAP Cloud Transport Management



What is the ABAP Cloud Development Model?

Cloud-Ready Development

The ABAP Cloud development model represents SAP's approach to cloud-ready ABAP development, designed specifically for extension development across all SAP S/4HANA versions and SAP BTP.

Strict Guidelines

This model enforces strict rules and guidelines to ensure extensions remain upgrade stable, cloud compatible, maintainable, and properly isolated from core functionality.

Future-Proof

By following this model, developers can create extensions that will continue to function properly through system upgrades and cloud migrations.

Key Components of the ABAP Cloud Development Model

1 Cloud-Optimized ABAP Language

ABAP Cloud is a subset of the traditional ABAP language, optimized for cloud environments. It restricts certain statements and constructs that could lead to upgrade issues, promotes clean coding practices, and enforces strict separation between custom code and SAP standard objects.

2 Prevents Direct Modifications

The model prevents direct modification of SAP standard code, ensuring that extensions remain isolated from the core system and reducing the risk of compatibility issues during upgrades.

3 Clean Architecture

By enforcing these restrictions, ABAP Cloud promotes a cleaner architecture that is more maintainable and less prone to technical debt over time.

Released APIs Only

Protected Access

Only SAP-released APIs and objects can be accessed, with access to unreleased objects blocked at both syntax and runtime levels.

Upgrade Stability

This ensures extensions remain stable during upgrades by preventing dependencies on internal SAP implementations that might change.

API-First Approach

Developers must adopt an API-first mindset, working only with officially supported interfaces rather than internal structures.

Integrated Development Environment

SAP Business Application Studio

A cloud-based development environment optimized for SAP extension development with built-in tools for ABAP Cloud.

Eclipse with ADT

Eclipse with ABAP Development Tools provides a comprehensive environment for ABAP development with syntax checking and validation.

Integrated Tools

Built-in tools for syntax checking and validation ensure code quality and compliance with ABAP Cloud guidelines.

Extension Components Library (XCO)

Reusable Components

The XCO library provides reusable components for common extension tasks, reducing development time and ensuring consistency.

General-Purpose Functionality

It offers general-purpose functionality that can be applied across different extension scenarios.

Modern API Design

XCO implements modern API design patterns that align with cloud development best practices.

Consistency

Using XCO ensures consistency across different extension scenarios and development teams.

Extension Patterns in ABAP Cloud

Business Object Extensions

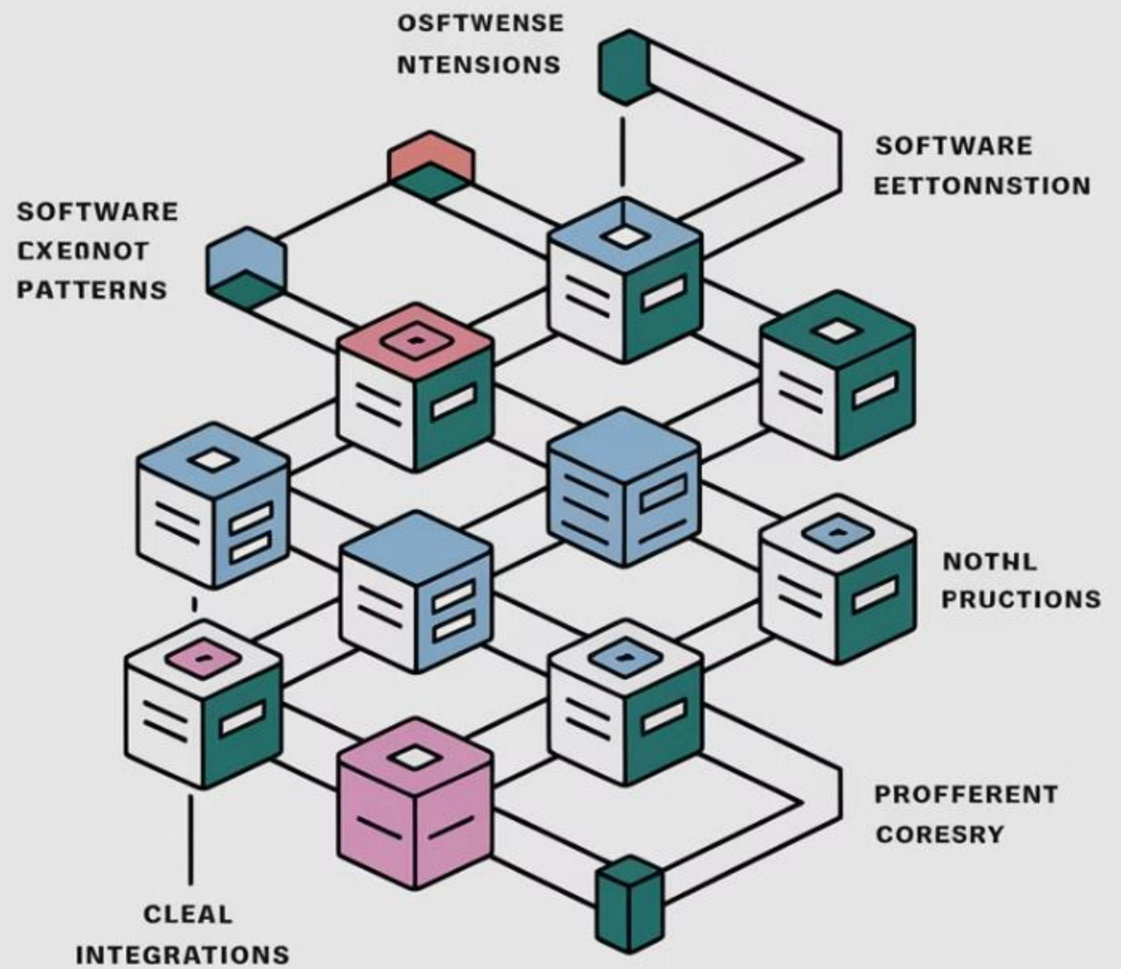
Custom fields can be added through key user extensibility, custom logic implemented through Business Event handling, and custom validations added at predefined extension points.

User Interface Extensions

UI adaptations can be made through key user tools, with custom UI elements added at designated extension points.

Process Extensions

Business processes can be extended using Business Events to trigger custom logic and custom steps added at designated process points.



User Interface Extensions

For extending SAP Fiori applications, UI adaptations can be made through key user tools, while custom UI elements can be added at designated extension points. Developers can create custom apps using SAP Fiori elements and consume released OData services for data access.

Process Extensions

1

Business Events

Business Events can trigger custom logic when specific actions occur in the standard system, allowing for non-disruptive extensions.

2

Process Points

Custom steps can be added at designated process points, extending the standard workflow without modifying core functionality.

3

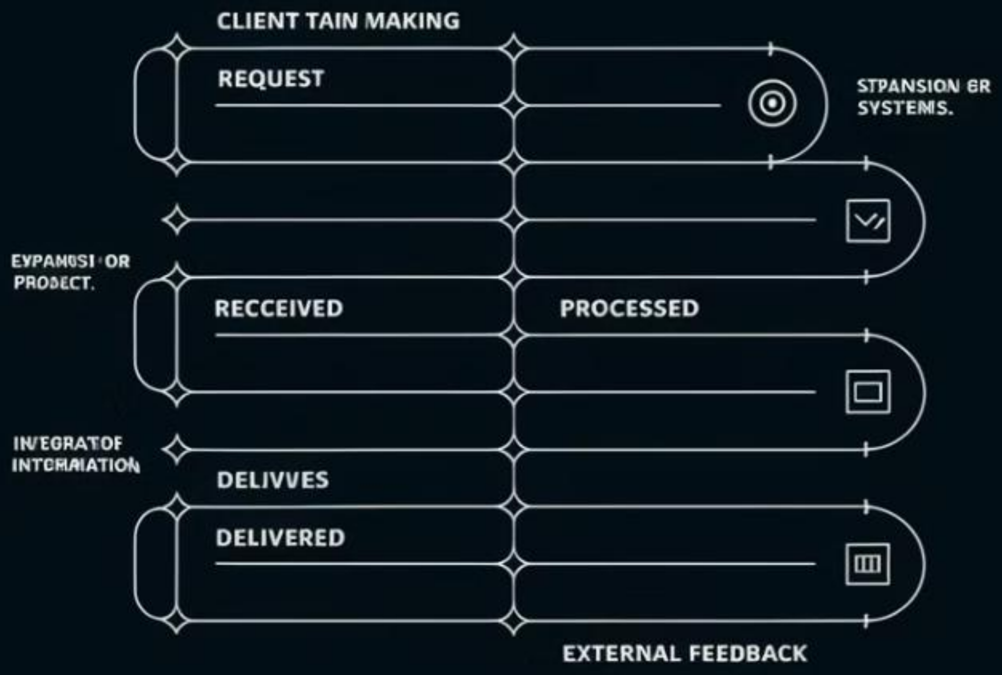
Workflow Extensions

Workflows can be extended using SAP Workflow Management, providing flexibility for complex business processes.

4

Integration

Integration with external systems can be built using released APIs, ensuring stable connections between systems.



Hypothetical Implementation Scenario

Business Need

A hypothetical manufacturing company needs to extend their SAP S/4HANA system to accommodate specialized quality management requirements, adding industry-specific quality parameters to materials and capturing additional data during quality inspection processes.

Traditional Approach (Not Recommended)

In the past, they might have modified standard SAP tables to add fields, added custom logic directly into standard SAP code, created Z-tables that mirrored SAP tables, and implemented implicit dependencies on internal SAP structures.

ABAP Cloud Approach

1 **Analyze Available Extension Points**

Review the standard quality management processes, identify SAP-provided extension points and Business Events, and determine which released APIs are available for use in the implementation.

2 **Implement Key User Extensions**

Add custom fields to material master and quality inspection screens, configure field properties and validations, and set up field visibility based on user roles.

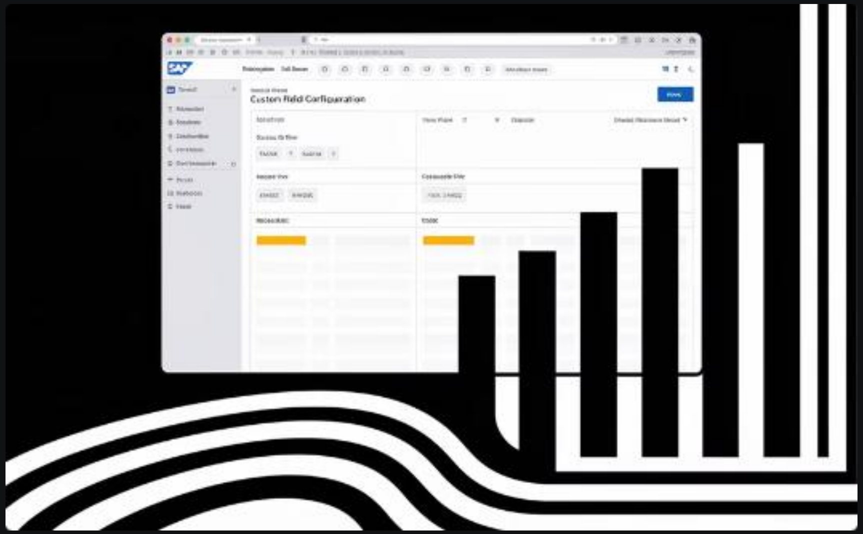
3 **Develop Custom Business Logic**

Create custom business objects for additional quality parameters and implement event handlers for standard quality management events.

4 **Build Custom UIs & Integrate**

Develop SAP Fiori applications and implement integration with external systems following clean core principles.

Implement Key User Extensions



Custom Fields

Add custom fields to material master and quality inspection screens to capture additional quality parameters required by the business.

Field Properties

Configure field properties and validations to ensure data quality and consistency across the system.

Role-Based Visibility

Set up field visibility based on user roles to ensure appropriate access to quality management information.

Develop Custom Business Logic

Custom Business Objects

Create custom business objects for additional quality parameters

Event Handlers

Implement event handlers for standard quality management events

RESTful APIs

Use the ABAP RESTful Application Programming Model

Testing & Validation

Ensure all custom logic works with standard processes

The ABAP RESTful Application Programming Model provides a framework for creating business applications and services that follow modern design principles while maintaining compatibility with SAP's cloud strategy.

Build Custom User Interfaces

1

Requirements Analysis

Identify user needs and workflow requirements

2

Fiori Elements Design

Use SAP Fiori elements with custom annotations

3

OData Integration

Connect to standard and custom OData services

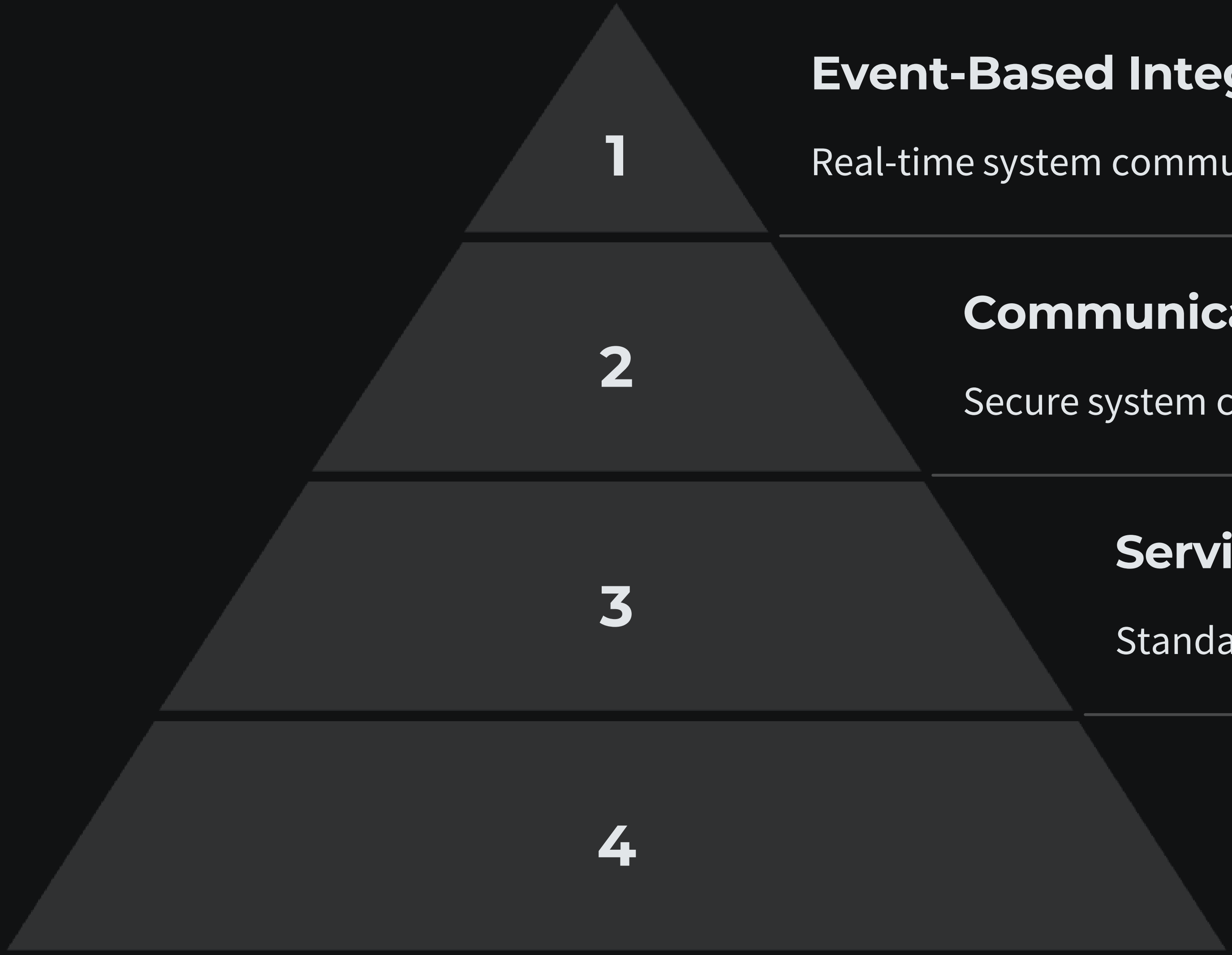
4

Testing & Deployment

Validate and deploy to production

SAP Fiori applications provide a modern, responsive user experience that works across devices while maintaining consistency with SAP's design language. By using Fiori elements, developers can create applications faster with less custom code.

Integrate with External Systems



Event-Based Integration

Real-time system communication

Communication Management

Secure system connections

Service Consumption Models

Standardized API usage

Clean Core Principles

Maintainable integration architecture

The result is a fully compliant extension that meets business needs while remaining upgrade-stable and cloud-ready. By following clean core integration principles, the solution avoids creating technical debt that would complicate future upgrades or cloud migrations.

Benefits and Limitations

Benefits

- Reduces technical debt through clean separation
- Ensures smoother upgrades with stable interfaces
- Facilitates cloud migration with compatible code
- Improves maintainability with standardized approaches
- Provides clear separation of concerns
- Enforces best practices automatically

Limitations

- Not all standard SAP functionality has released APIs yet
- Some complex scenarios may require Tier 2 approaches
- Legacy extensions may need significant refactoring
- Learning curve for developers accustomed to traditional ABAP

Limitations to Consider

70%

API Coverage

Not all standard SAP functionality has released APIs yet, which may limit what can be accomplished with the ABAP Cloud development model in certain scenarios.

25%

Complex Scenarios

Some complex scenarios may require Tier 2 approaches when the limitations of released APIs prevent implementing all requirements within Tier 1.

100%

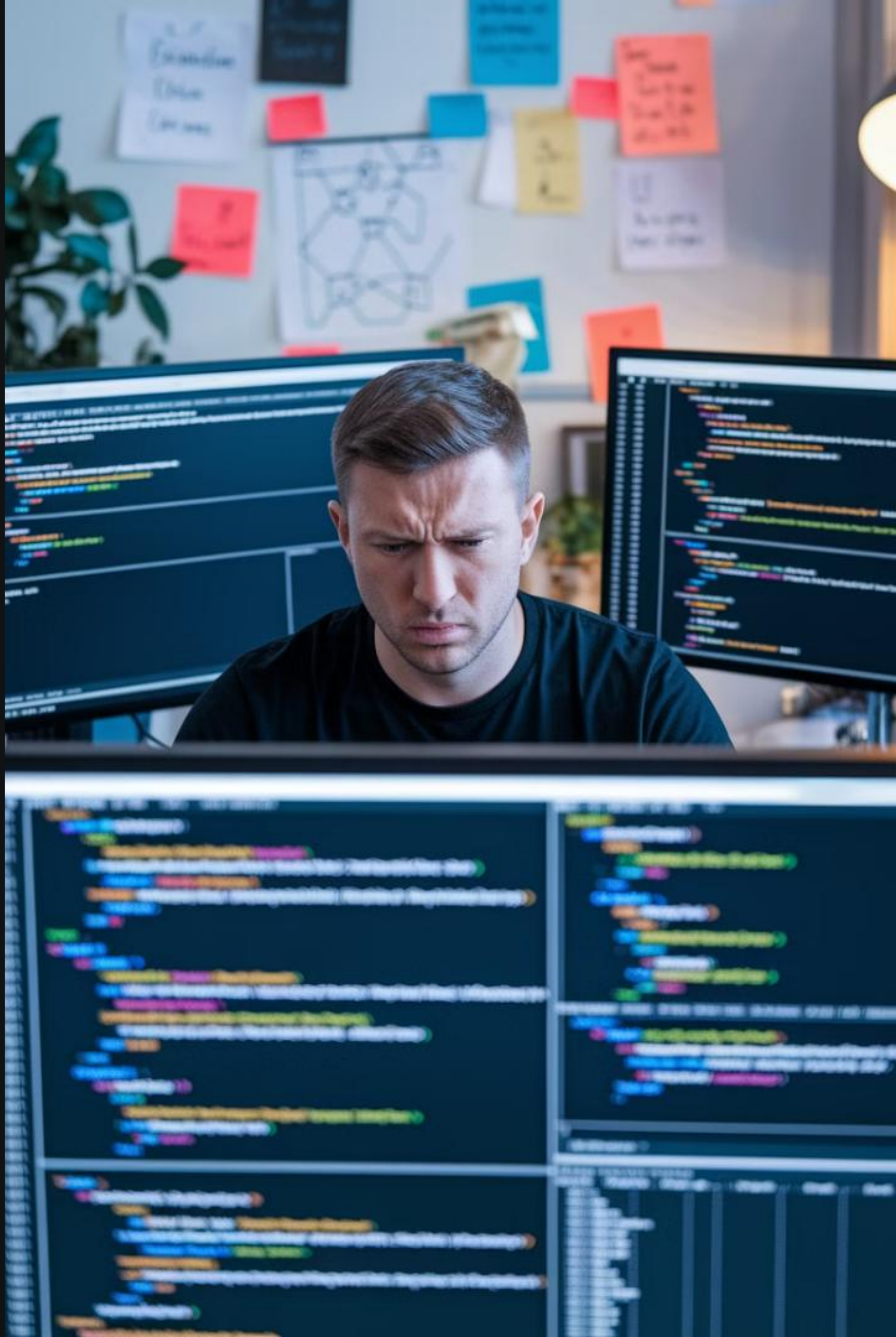
Legacy Refactoring

Legacy extensions may need significant refactoring to comply with ABAP Cloud principles, potentially requiring substantial investment.

3-6

Learning Curve

Developers accustomed to traditional ABAP face a learning curve when adapting to the restrictions and patterns of the ABAP Cloud model.



Governance and Best Practices

1

Establish Clear Extension Governance

Create policies and guidelines that define how extensions should be developed, reviewed, and maintained within your organization to ensure compliance with ABAP Cloud principles.

2

Train Developers

Invest in comprehensive training for developers on ABAP Cloud principles, ensuring they understand both the technical restrictions and the architectural benefits of this approach.

3

Review Extension Designs

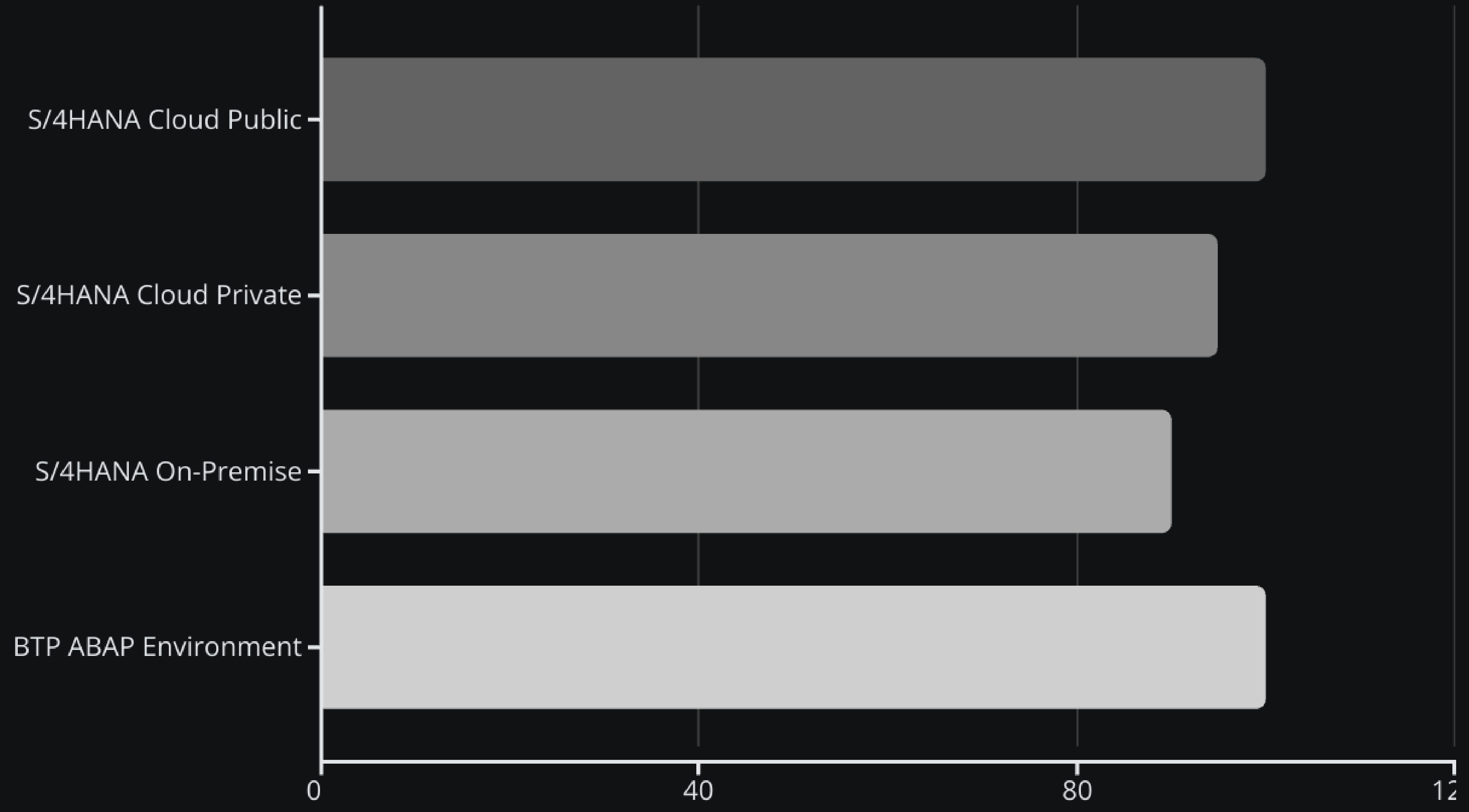
Implement a design review process where extension architectures are evaluated before implementation begins to identify potential issues early.

4

Document and Assess

Maintain thorough documentation of all extensions and regularly assess them for compliance with best practices and cloud readiness.

Compatibility Across SAP Environments



An important advantage of the ABAP Cloud development model is its compatibility across different SAP environments. This enables a "develop once, run anywhere" approach for extensions, saving time and resources while ensuring consistency across deployment scenarios.

Extensions developed following ABAP Cloud principles can work in SAP S/4HANA Cloud Public Edition, SAP S/4HANA Cloud Private Edition, SAP S/4HANA On-Premise, and SAP BTP ABAP Environment with minimal adjustments.

Conclusion

Fundamental Shift

The ABAP Cloud development model represents a fundamental shift in how we develop extensions for SAP systems, moving from direct modifications to API-based extensions.

Future-Proof Extensions

By embracing this model, organizations can build future-proof extensions that remain stable through upgrades and cloud migrations, reducing long-term maintenance costs.

Clean Core Strategy

This approach aligns with SAP's clean core strategy, ensuring that customizations remain separate from standard functionality while still meeting business requirements.

Next Steps: RICEFW Implementation Framework

Reports

Custom reporting solutions that leverage standard SAP data models through released APIs while maintaining upgrade compatibility.

Interfaces

Integration approaches that connect SAP systems with external applications using cloud-ready technologies and patterns.

Workflows

Process automation and extension techniques that enhance standard SAP processes without modifying core functionality.

Forms

Document generation and management solutions that work seamlessly across cloud and on-premise environments.

In our next lesson, we'll explore the RICEFW implementation framework and how it applies to modern SAP extension development, building on the ABAP Cloud principles we've discussed today.