Custom Extensions in SAP S/4HANA® Implementations
A Practical Guide for Senior IT Leadership
March 2020
# Table of Contents

3 Forewords by DSAG, Gunther Rothermel, and Bjoern Braemer

6 Introduction – The (R)Evolution of Enterprise Applications

7 Part One – Key Concepts for Custom Extensions in SAP S/4HANA

7 Key Takeaways

9 Consumer-Grade UX

10 Clean Digital Core

11 Extensibility Framework of SAP S/4HANA

12 In-App Extensibility

14 Side-by-Side Extensibility with SAP Cloud Platform

20 DevOps

21 SAP Cloud SDK Continuous Delivery Pipeline

23 Part Two – Custom Code in System Conversions

23 Key Takeaways

24 Removing Obsolete Custom Code

29 Automated and Manual Custom Code Adaptation

35 Review of Modifications, Clones, and Implicit Enhancements

37 Performance Optimization of Custom Code

38 Rethink, Not Just Rework

40 Part Three – Custom Extensions in New Implementations

40 Key Takeaways

41 Keep the Core Clean

42 Understand the New Technologies

44 Avoid Lift and Shift

45 Starting with SAP Cloud Platform

48 Part Four – Key Architectural Patterns and Decision Matrix

49 Pattern #1: Cloud Application

51 Pattern #2: Hybrid Application

53 Pattern #3: In-App Extensibility

55 Pattern #4: Classical Extension

56 Decision Matrix

57 Additional Considerations

58 Conclusion

59 Acknowledgments

60 Glossary
Foreword by DSAG

Custom extensions for SAP S/4HANA® can accelerate the digital transformation of companies and enrich their IT landscapes. Therefore, DSAG welcomes this guide, which offers valuable support and practical tips for SAP customers who are dealing with extensions in their SAP S/4HANA projects.

This document provides developers, administrators and project teams with an overview of how to decouple applications from a legacy system and integrate them as functionality into a new one. This guide explains the key concepts of building custom extensions in SAP S/4HANA as well as those of SAP® Cloud Platform. In addition, it delivers useful guidance on the various options available for dealing with custom code.

As DSAG, we understand and appreciate the added value of this guide, especially when used in combination with Mapping Your Journey to SAP S/4HANA – A Practical Guide for Senior IT Leadership, SAP’s handbook for planning a successful migration. Together, both guides can make it much easier for your organization to move to SAP S/4HANA.

Ralf Peters
DSAG Board Member
Digitization, Finance, and Value Chain
The SAP HANA® platform opened up new frontiers for what business applications could achieve in a data-driven world. SAP S/4HANA is a demonstrated manifestation of this potential. It provides a gateway to vast business networks and processes that used to stop at the walls of a company. When used together with SAP Cloud Platform, SAP S/4HANA helps you create infinite opportunities for your business.

SAP Cloud Platform is the integration and extension platform at SAP. It provides enterprises with flexibility and speed in their continuous quest for competitive differentiation in a constantly changing world. SAP Cloud Platform and its intelligent technologies, such as machine learning, robotic process automation, and conversational artificial intelligence, help strengthen and streamline your internal and external processes – resulting in improved experiences for your employees, customers, and partners. A modular, step-by-step approach enables you to transition to the cloud at your pace. Extensions are created on top of SAP Cloud Platform to keep your digital ERP core clean while increasing the agility and vigor of your business.

This document comprehensively outlines the rich extensibility capabilities of SAP S/4HANA that you can use to differentiate your business processes and applications. It also explains how SAP Cloud Platform complements these capabilities with its own extensibility functionality. By combining the extensibility of SAP S/4HANA and SAP Cloud Platform, you can easily extend our intelligent suite with your own unique and innovative capabilities and simultaneously integrate them with third-party solutions. The result is the ability to build new, high-impact applications and extensions that optimize your business processes and your success.

Gunther Rothermel
Executive Vice President
Head of SAP Cloud Platform at SAP SE
adapt existing solutions, and develop new ones to meet evolving requirements. SAP’s latest product portfolio – built around SAP S/4HANA as the new digital ERP core – enables our customers and partners across industries to address their changing business requirements and meet the challenges of the future.

This document provides best practices for your transformation journey to a clean, digital SAP S/4HANA system. It offers recommendations for leveraging custom code and new technologies that can help your company maintain competitive advantage through the individualization of applications. Change is necessary to thrive in the digital era. The best practices outlined in this document are designed to help you become an effective change agent and streamline your company’s digital transformation journey.

Bjoern Braemer
Senior Vice President
Global Head of the SAP S/4HANA Movement Program at SAP SE

“Where would you like to see your company 5 or 10 years from now?”

It’s the question I ask our customers when we start the conversation about their transformation journey toward the new digital age.

It is essential that IT and business leaders think about their company’s long-term strategy and compare their current operations model with future requirements that are being driven by ever-changing customer expectations, competition, regulations, and other market forces. To keep pace in such a dynamic environment, companies must encourage and absorb change in short cycles. They must continuously adapt their business models by leveraging a flexible infrastructure that enables them to rapidly adjust key applications.

Developing solutions ahead of immediate demand is a core competency of successful software vendors. So, too, is delivering a comprehensive platform that allows customers to quickly and continuously consume the latest innovations.
Introduction

THE (R)EVOLUTION OF ENTERPRISE APPLICATIONS

In 1992, the idea of disclosing the source code of a commercial application – combined with the ability for customers to extend and even modify this code in a controlled manner – was way ahead of the IT industry’s mainstream.

During the decade that followed, it turned out to be one of the decisive advantages of SAP R/3® software – just like its client-server architecture, logical unit of work (LUW) concept, and ability to store code along with customizing and application data in the same database. In hindsight, SAP’s success story would have been unthinkable without SE38.

Three decades and a few billion lines of code later, SAP’s vision of how to enable customers to extend enterprise applications is far more complete. It primarily relies on four key concepts: consumer-grade user experience (UX), clean core, extensibility framework, and cloud development.

This document is about what these concepts mean in practice and how to apply the associated new tools and technologies in your SAP S/4HANA® project. A team of SAP’s subject-matter experts, developers, and consultants composed it as a practical, down-to-earth guide to help you as an IT leader effectively prepare for the transformation ahead and employ the new SAP technologies for your business.

This document has four parts. “Part One” introduces you to the essential concepts for a modern enterprise application’s extensibility. “Part Two” guides you through the key aspects of dealing with your custom code during a system conversion. “Part Three” offers practical advice for those who run a new implementation of SAP S/4HANA or launch new SAP technologies in their organization. Finally, “Part Four” introduces the architectural patterns you are most likely to lean on when building custom extensions for SAP S/4HANA and gives guidance on how to choose between them.

A TIME JOURNEY

1992
SAP releases SAP® R/3® software

1995
SAP unveils SAP Cloud Platform

1996
Sun Microsystems releases Java 1.0

1998
The term “open source” emerges

1999
SOAP standard is published

2007
Microsoft launches OData

2011
SAP launches the SAP HANA® database

2012
SAP launches SAP S/4HANA®

2015
SAP unveils SAP Cloud Platform initially as SAP HANA Cloud Platform

© 2020 SAP SE or an SAP affiliate company. All rights reserved.
Part One

KEY CONCEPTS FOR CUSTOM EXTENSIONS IN SAP S/4HANA

KEY TAKEAWAYS

• Enterprises should take the user experience seriously. Well-designed applications can significantly improve users’ productivity and change ways of working.

• SAP Fiori® is a design system that includes visual design, a navigation concept, interaction patterns, built-in adaptive rendering, an application provisioning concept, and more.

• SAP’s rationale behind the “clean digital core” paradigm is simple: allow customers to extend their SAP S/4HANA software while making the eventual updates non-events.

• In-app extensibility is a very powerful tool that allows you to implement the typical extension in SAP S/4HANA – mostly without a single line of code.

• SAP Cloud Platform allows you to build larger, decoupled, side-by-side custom extensions.

• The extensibility framework of SAP S/4HANA unites these two ways of extending applications into a single framework. “In-app extensibility” and “side-by-side extensibility” are mostly used in combination.

• The programming language you choose for your extension – ABAP®, Java, or JavaScript – is decisive. Not only does it determine your development tools, but it also implies the programming model of your application.

• True decoupling between your custom applications and SAP S/4HANA enables an independent lifecycle and allows you to build and evolve applications much faster.

• DevOps is a set of practices spanning and building a closed loop over all stages of the software lifecycle, from development to operations. SAP offers an embedded continuous delivery pipeline for SAP Cloud Platform.
This section explains the four concepts that are key for custom extensions in SAP S/4HANA: the consumer-grade user experience (UX), clean core, extensibility framework including in-app extensibility and side-by-side extensibility, and cloud development (see Figure 1). The latter means a development platform in the cloud, the proposed application programming models, and the way in which development on this platform is carried out.

These concepts are interrelated and rely on each other: a clean core is impossible without a powerful extensibility framework; likewise, creating a consumer-grade UX makes modern programming models and tools indispensable.

In this section, we explain how SAP realizes these concepts in practice.

---

**Figure 1: Key Concepts for Custom Extensions in SAP S/4HANA**

Cloud development

Platform + programming model + DevOps

Clean core

Consumer-grade UX

Extensibility

In-app + side-by-side
CONSUMER-GRADE UX

The digitalization of society has profoundly changed the way users think of enterprise applications. The younger generation starting their career in your company expects the same level of comfort they know from consumer-grade applications or Web sites. They expect enterprise applications to be visually attractive, accessible from any device at any time, simple to use, responsive, and easily personalized.

However, pleasing users is not the main reason why enterprises should take UX seriously. Well-designed applications:

• Require less time to learn
• Result in fewer errors made by users
• Give users fewer reasons to work outside of the system
• Eliminate confusion by following the “principle of one” and offering a single application for a particular purpose
• Can significantly improve productivity and change ways of working

SAP’s front-end technologies for building applications to the above standards are as follows:

• SAPUI5 is SAP’s main UI framework. It is built on HTML5, CSS3, JavaScript, LESS, and jQuery standards and enhanced to match the needs of enterprise application development.
• SAP Fiori is a design system that includes visual design, a navigation concept, interaction patterns, an application provisioning concept, and more. SAP Fiori elements (formerly known as smart templates) provide a framework for the most common application patterns.
• Other purpose-built tools include SAP S/4HANA embedded analytics and the SAP CoPilot digital assistant, for example.

SAP has been continuously developing new SAP Fiori apps while visually harmonizing traditional SAP GUI interface transactions and Web Dynpro ABAP applications with the SAP Fiori visual theme. For an excellent introduction to the concepts of SAP Fiori, please click here. More information on SAPUI5 is available here.

DID YOU KNOW?
The SAP Fiori® UX has received numerous design recognitions from some of the most prestigious organizations in the world:

• Red Dot Award (Next-generation SAP Fiori UX design concept)
• Red Dot Award (SAP Fiori UX embedded analytics concept)
CLEAN DIGITAL CORE

Every aspect of global business now seems to move significantly faster than it did even 10 years ago. For on-premise enterprise systems, a clean digital core is the only way to cope with that speed.

SAP’s rationale behind the “clean digital core” paradigm is simple: allow customers to extend their SAP S/4HANA software while making the software updates eventually non-events.

A clean core allows faster software deployment as well as easier adoption of both SAP innovations and the regulatory changes to software. For IT service providers, it means being able to offer upgrade projects at a fixed price. For your business, it means a reduction in total cost of ownership (TCO) without jeopardizing flexibility.

In practical terms, keeping the core clean means:
- Applying a zero-modification policy from the project’s first day
- Eliminating enhancements that are redundant to standard code and functionality, as well as “clones” of standard code
- Using white-listed APIs

- Leveraging the in-app extensibility of SAP S/4HANA to its full extent
- Employing the capabilities and services offered by SAP Cloud Platform to build larger extension applications
- Utilizing the SAP Cloud Platform Integration service

For converted systems, cleaning up the core means:
- Removing obsolete custom code, that is, the code that is either not used anymore or can be replaced with standard or partner solutions
- Reviewing modifications and clones
- Replacing outdated technologies

For smaller systems, this can be done in one attempt. However, teams dealing with large systems often choose another strategy: apply the clean core guidelines for every custom development that needs to be adapted.

Read more about how to make a clean digital core attainable in practice in “Part Two” and “Part Three” of this guide.
EXTENSIBILITY FRAMEWORK OF SAP S/4HANA

Any veteran SAP consultant would probably agree that:

- The most frequent types of extensions in the SAP ERP application require a very limited set of features.
- Many custom ABAP programs need very little of the core ERP data and mostly access it in read-only mode.

The first insight resulted in what we call “in-app extensibility” and the second in “side-by-side extensibility.” SAP has united these two ways of extending its applications into the single extensibility framework of SAP S/4HANA (see Figure 2). As a matter of fact, both types of extensibility are mostly used in combination; therefore, it is almost never an either/or decision.

The extensibility framework of SAP S/4HANA simplifies and speeds up the development of custom extensions by automating the common steps and offering a standardized architecture.

We explain both types of extensibility in the next sections.

Figure 2: Extensibility Framework of SAP S/4HANA®
IN-APP EXTENSIBILITY

The idea of in-app extensibility builds on the insight that most ERP application extensions are very similar in nature and, in fact, require a limited set of technologies. The in-app extensibility mechanisms in SAP S/4HANA have been designed to enable the key users to do such extensions in a self-service fashion. They offer a very powerful tool set to extend the data model, business logic, and UIs of SAP S/4HANA.

SAP has defined eight in-app extensibility patterns (see Figure 2):

- **Adopt existing SAP Fiori UIs** – Add, hide, move, or regroup fields on screen, add custom fields, and change label texts.
- **Add custom business logic** – Extend your business processes by implementing business add-ins (BAdIs).
- **Create custom fields** – Extend standard business objects (for example, material master) by adding custom fields and processing logic. When you add a field, it becomes visible in all related standard core data services (CDS) views and OData services and, thus, also becomes available in the corresponding SAP Fiori apps.
- **Expose the data** – Based on standard white-listed CDS views, you can create your own custom CDS views and external read-only APIs to enable external data consumption.
- **Create custom forms or e-mail templates** – Customize new printed and e-mail forms (using Adobe LiveCycle Designer on premise or SAP Cloud Platform) based on an existing data source, and extend your forms and e-mail templates with new custom fields.
- **Create custom analytics** – Create custom CDS views to build queries with the “Custom Analytical Queries” SAP Fiori app, or develop real-time analytics and KPI tiles with SAP Smart Business cockpits.
- **Create custom business objects** – Create your own business objects and custom tables with cross-references, including automatically generated maintenance UIs and OData services.
- **Create custom UIs** – Create your own custom user interface for your custom business objects using SAP Web IDE running on SAP Cloud Platform, and deploy it to your SAP S/4HANA Cloud tenant.

Watch this demo video to see how powerful these no-code adjustments can be.

In-app extensibility and side-by-side extensibility are mostly used in combination; it is almost never an either/or decision.
Much like customizing, these adaptations are made in the development (DEV) system and transported to production through a transport request. In SAP S/4HANA Cloud, you create them in the quality assurance (QA) tenant and push to the production (PROD) tenant. You can use the built-in extensibility inventory to get an overview of all in-app enhancements made.

Please visit the [SAP Extensibility Explorer](https://www.sap.com) site or read the excellent blog titled “[The Key User Extensibility Tools of SAP S/4HANA](https://www.sap.com)” to learn more about the extensibility patterns and the technology behind them.

This discussion wouldn’t be complete without a few words on the traditional extensibility options in SAP ERP. User exits, enhancement points, and other extensibility techniques available in your SAP ERP system will continue to work after it has been converted to SAP S/4HANA. Of course, you should avoid using them in the future.

In-app extensibility is a very powerful tool that allows you to implement **typical extensions** in SAP S/4HANA – mostly without a single line of code.
SIDE-BY-SIDE EXTENSIBILITY WITH SAP CLOUD PLATFORM

SAP Cloud Platform has been designed to make it much easier to build and integrate side-by-side extensions for SAP products compared to any other generic platform. Consider these traits:

- SAP Cloud Platform is one platform for extending both cloud and on-premise solutions from SAP.
- It provides integration for your extensions on the UX, process, and data levels.
- It offers full end-to-end security from the UI to the back end.
- It enables smooth connectivity to SAP solutions and easy discovery and consumption of APIs and events across all applications.
- It is also integrated into the lifecycle management for hybrid landscapes.

SAP Cloud Platform offers a wide range of capabilities and services tailored to the needs of enterprise applications. Explaining all of these is beyond the scope of this document. However, we would like to make you familiar with the most fundamental concepts for side-by-side extensibility scenarios with SAP S/4HANA (see Figure 3).

Connect and Communicate

As part of the SAP Cloud Platform Connectivity service, SAP offers a cloud connector as a simple and secure way to connect your on-premise landscape to SAP Cloud Platform.

The connection between your SAP Cloud Platform account and your cloud solutions from SAP (for example, SAP SuccessFactors® solutions or the SAP Cloud Platform Integration service) goes directly through a secure Internet channel.

Establishing such connections has been greatly simplified through the extensibility services to connect systems to SAP Cloud Platform. The number of manual steps (such as exchanging certificates or entering URLs) has been massively reduced. It now takes just a few clicks.¹

---

¹ Enabled for SAP S/4HANA Cloud and SAP SuccessFactors solutions; other solutions are on the road map.
Figure 3: Side-by-Side Extensibility with SAP® Cloud Platform

- SAP S/4HANA®
- SAP® ERP Central Component®
- SAP C/4HANA® suite
- SAP SuccessFactors® solutions
- SAP Concur® solutions*
- SAP Fieldglass® solutions*
- SAP Ariba® solutions*
- Other SAP solutions*
- Third-party solutions*

- Connectivity (including cloud connector)
- Extensibility services
- SAP Fiori® launchpad
- Identity and access management
- SAP API Management

- SAP Cloud Platform Integration service
- Business event bus (SAP Cloud Platform Enterprise Messaging)
- Mendix
- Robotic process automation
- Workflow
- Business rules
- SAP Business Application Studio
- ABAP® development tools (ADT) for Eclipse
- ABAP® RESTful Programming Model
- SAP Cloud SDK

- SAP API Business Hub
- SAP Graph
- Build
- Application runtime
- Serverless runtime
- Kyma® runtime
- ABAP environment

- Life cycle (continuous integration and continuous delivery)
- Extension center
- Monitoring
- Landscape management
- Operate

*This is the current state of planning and may be changed by SAP at any time without notice.
Once your SAP solutions have been connected to SAP Cloud Platform, the APIs and events exposed by these solutions become visible within the assigned subaccounts, so the developers can easily discover and consume them (see Figure 4).

Another communication style that the platform offers for Java and JavaScript applications is event-based communication, also known as event bus. SAP S/4HANA, SAP S/4HANA Cloud, SAP SuccessFactors solutions, the SAP C/4HANA® suite, and the SAP Sales Cloud and SAP Service Cloud solutions natively support application events. These events can be published through the SAP Cloud Platform Enterprise Messaging service, where applications can subscribe to these events. For instance, an application can subscribe to the creation of a new business partner in SAP S/4HANA and will be notified whenever a new business partner is created.

The platform supports publish-subscribe, point-to-point, and request-reply communication patterns. Other applications can define their own events using SAP Cloud Application Programming Model.

SAP API Business Hub offers detailed documentation on all available APIs and events and allows users to try them out directly in their browser.
Build Your Extensions
There are three options for extending your enterprise applications:

- Leverage default application programming models (see below)
- Employ low-code and no-code tools (for example, workflow, business rules, Mendix)
- Use open source components or external frameworks of your choice

While the second and third options are self-explaining, we’d like to elaborate on the first one.

An application programming model is a set of guidelines prescribing how an application is supposed to be built. SAP Cloud Platform defines one programming model for applications written in Java or JavaScript (called SAP Cloud Application Programming Model) and a second one for applications written in ABAP (called ABAP RESTful\(^2\) Programming Model).

These two models have a lot in common but also have some differences, as shown in the following table.

---

2. Representational state transfer (REST) is an architectural style for creating Web services as a set of stateless operations. Such Web services are called RESTful services.
### SAP® Cloud Application Programming Model Versus ABAP® RESTful Programming Model

<table>
<thead>
<tr>
<th>Common to Both Models</th>
<th>ABAP® RESTful Programming Model</th>
<th>SAP® Cloud Application Programming Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>• RESTful OData services for communication between the front end (UI) and the application on SAP® Cloud Platform</td>
<td>• Supports the ABAP programming language</td>
<td>• Supports Java or JavaScript (node.js)</td>
</tr>
<tr>
<td>• Core data services (CDS) language to define the data model, business objects, and their attributes and behavior, and the generation of RESTful OData services from these definitions</td>
<td>• Provides Git-enabled lifecycle management</td>
<td>• Enables applications originally written as single-tenant applications to be turned into multitenant ones through configuration</td>
</tr>
<tr>
<td>• The universal nature of CDS views, which can be connected to any data source to fetch or change the data: SAP HANA® database, API of an SAP solution, or third-party REST service</td>
<td>• Offers a service consumption model for easy remote OData service calls</td>
<td>• Leverages event-based communication using the SAP Cloud Platform Enterprise Messaging service</td>
</tr>
<tr>
<td>• Built-in extensibility capabilities, enabling users to extend both SAP Cloud Application Programming Model and ABAP® RESTful Programming Model in a similar way as with the in-app extensibility of SAP S/4HANA® (see above)</td>
<td>• Enables the possibility to reuse selected custom code in the cloud with SAP Cloud Platform, ABAP environment, while rebuilding the UI and back-end access</td>
<td>• Wraps the REST service calls to the underlying back-end system to Java or JavaScript functions using SAP Cloud SDK</td>
</tr>
</tbody>
</table>

© 2020 SAP SE or an SAP affiliate company. All rights reserved.
**Provision As an Application or As a Web Service**

With either application development model, you can provision your extension as a visual application or as a RESTful OData service.

For applications, there are no restrictions on the UI technology. However, SAP Fiori elements are strongly preferred because of their native support for CDS annotations (metadata that describes how the data should be displayed).

If you expose your extension as a service for other internal applications or externally, the SAP Cloud Platform API Management service provides you with all necessary control over API access and usage.

More information is available at:

- “SAP S/4HANA and SAP S/4HANA Cloud Extensibility for Customers and Partners” white paper
- SAP Cloud Platform capabilities
- SAP Extensibility Explorer site
- SAP Cloud Platform Discovery Center site
- “Create and Deliver Cloud-Native SAP S/4HANA Extensions” course on the openSAP platform
- “Extending SAP S/4HANA with SAP Cloud Platform” openSAP course
- “SAP Cloud Platform Essentials” openSAP course
- SAP Cloud Platform trial
- Customize Your SAP S/4HANA System with SAP Cloud Platform book
DevOps is a set of practices spanning and building a closed loop over all stages of the software lifecycle, from development to operations. Thus, it subsumes the ideas of agile development, continuous integration, continuous delivery, and continuous deployment (see Figure 5).

In a way, it responds to the challenges associated with the increased speed of business by drastically shortening the idea-to-solution cycle. At SAP, we see DevOps as a combination of:

- Small (6 to 10 experts), independent, self-sufficient development teams distributed across your enterprise that build custom applications to support your business
- An automated process of building, testing, and deploying your applications – allowing up to multiple cycles a day
- Enforcement of code quality through a series of code checks and security scans that block the application’s deployment if errors are found
- Central orchestration of the pipeline to ensure all development teams comply with the common quality and security guidelines

More information about DevOps is available at SAP Community.

We also highly recommend the State of DevOps Report by DevOps Research and Assessment LLC (DORA) to the readers looking for actionable guidance on how to improve their software delivery performance.

---

**Figure 5: The Core Values of DevOps**

**Trust**
The approval processes are known to extend development times, so cutting them out allows much shorter delivery cycles.

**One team**
Having all experts in one team – developers, UX designers, QA experts, and IT operations specialists – results in quality and efficiency.

**Automation**
The fewer manual actions between software development and release, the better.

**Code quality**
Frequent code checks, security scans, and automated testing aim at “failing fast” and fixing bugs at an early stage.
Setting up your own continuous delivery pipeline can be labor intensive and prone to errors. Instead, SAP Cloud SDK provides a prebuilt continuous delivery stack based on an open-source automation server called Jenkins. This stack can be easily instantiated on any machine equipped with Docker, or in a Kubernetes cluster.

With SAP Cloud SDK, SAP offers a prebuilt continuous delivery pipeline for SAP Cloud Platform (see Figure 6).

---

**Figure 6: Prebuilt Pipeline in SAP® Cloud SDK**

- Start
- Init
- Build
- Local tests
  - Backend integration tests
  - Backend unit tests
  - Frontend integration tests
  - Frontend unit tests
  - Lint
  - NPM dependency audit
  - Static code checks
- Remote tests
  - End-to-end tests
  - Performance tests
- Quality checks
- Third-party checks
  - Additional tools
  - Checkmarx scan
  - Fortify scan
  - SonarQube scan
  - SourceClear scan
- Artifact deployment
- Production deployment
- End
The SAP Cloud SDK pipeline builds on SAP’s best practices and leverages SAP’s experience in building and delivering enterprise cloud applications. It allows you to:

- Detect bad coding practices through static code checks and discover inapt usage of SAP libraries or services (for instance, usage of non-white-listed SAP S/4HANA APIs)
- Detect code vulnerabilities\(^3\)
- Build the cloud application in a consistent manner
- Help ensure the application’s functional correctness through unit, integration, and end-to-end tests
- Maximize the application’s responsiveness with the help of performance tests
- Detect nonresilient practices of integrating remote systems and services
- Deploy the application to an artifact repository for audit purposes
- Deploy the successfully validated application to its production environment in a nondisruptive manner

You can adopt the SAP Cloud SDK pipeline without writing a single line of code. For specific requirements, it can be flexibly extended – without losing any benefits or provided capabilities.

The continuous delivery pipeline in SAP Cloud SDK is aligned with SAP’s certification processes for cloud applications. If you plan to certify your cloud application by SAP, the built-in automated quality reporting will make the certification process easier.

Customers and partners within the SAP ecosystem are advised to leverage these pipeline tools from day one of their project to facilitate the DevOps principles through continuous delivery and deployment.

More information is available [here](#).

\(^3\) You may choose to use third-party pluggable tools to achieve an even higher level of security.
Part Two

CUSTOM CODE IN SYSTEM CONVERSIONS

KEY TAKEAWAYS

• Consider system conversion as a once-in-a-lifetime opportunity to simplify and clean up your solution.

• Achieving a clean core with system conversion is challenging. Nevertheless, invest in the cleanup of modifications (both direct and indirect) and obsolete custom developments that are either not used or can be replaced with standard or partner solutions. This will pay off during the subsequent business transformation and future upgrades.

• Removing obsolete code significantly reduces the effort for custom code adaptation.

• All custom code that you keep will have to be adopted to the code line of SAP S/4HANA.

• With the quick fixes delivered by SAP as part of ABAP development tools, you can expect an automation rate of around 60%.

• Adoption of the new technology and tools by the development team is a key success factor.

In conversion projects, SAP suggests organizing the work into four work packages:

• Removal of obsolete custom code that is either unused or gets replaced with standard functionality as part of the conversion

• Automated and manual code adaptation

• Review of modifications, clones, and implicit enhancements

• Performance optimization

Projects that follow this recommendation can expect significantly less effort, a smooth execution, and a much cleaner system as a result (see Figure 7).

Figure 7: Exemplary Calculation for 1,000 Findings in Custom Code
REMOVING OBSOLETE CUSTOM CODE

Our statistics from the past 15 years show that 30% to 60% of custom code in an SAP ERP system is not executed. Another significant part can get replaced with SAP standard code upon system conversion. Thus, removing obsolete code significantly reduces the effort for custom code adaptation. At the same time, this is certainly a first step toward a clean digital core.

Building a business case for obsolete code removal initiatives was a challenge in the past, primarily because of the associated testing effort and the difficulty in measuring the business value or savings. Both these factors change in an SAP S/4HANA conversion. First, you can show immediate savings because you don’t have to adopt the deleted code. Second, the safety concerns (“Will it still run smoothly after we delete what we ostensibly don’t use?”) are addressed by the comprehensive functional tests executed by your team as part of the conversion.

If you decide against deleting obsolete code, you will need to adopt it too. Otherwise, you will introduce the risk of system dumps and even data inconsistencies.

SAP has made the deletion procedure an integral part of the conversion and has enabled it through integration between the “Custom Code Migration” (CCM) SAP Fiori app in SAP S/4HANA and the Software Update Manager tool.
Technically, the procedure includes the following steps:

- Collecting usage data – This is done through activating Usage and Procedure Logging (UPL) or ABAP Call Monitor (SCMON for short) in the production instance of your SAP ERP.
- Defining the scope – Based on the usage data, the CCM app proposes a set of objects that can be safely deleted. You can refine this proposal manually in your SAP S/4HANA sandbox.
- Preparing a backup and defining the restore procedure.
- Testing the deletion in a sandbox
- Deleting the objects during the system conversion of your development system

These steps are illustrated in Figure 8.

**Collect Usage Data**

To collect code usage information on the object level, you can leverage either UPL or its successor, SCMON. The latter is generally preferable, and its main advantage is the aggregation of the usage data directly in the SAP ERP system (SUSG transaction). However, there is no reason to change the setup if you already have UPL or SCMON activated with data accumulation in SAP Solution Manager.4

---

4. Another key difference between UPL and SCMON is the way in which execution statistics are stored. While UPL keeps all executed ABAP objects in a single flat list, SCMON keeps separate logs for each system entry point such as transactions, reports, and so on. SCMON is more useful for an in-depth analysis of custom code.
Both technologies are integrated into the kernel of the SAP NetWeaver® technology platform, so the performance impact is hardly noticeable. Having activated this function, you would usually collect the data for 6 to 18 months, including the year-end closing, to build confidence. Therefore, we suggest putting the below items onto your next-action list:

- If neither UPL nor SCMON is already active in your system, activate SCMON to enable the data aggregation option in your production SAP ERP using the transaction SUSG.
- Make sure that SCMON is configured for continuous data collection for at least 365 days and data volume limits are set high enough to avoid auto-deactivation.

• With either option, monitor the collectors and check the logs at regular intervals.

More information is available in the following documents:

• **Aggregate Usage Data in Your Production System with SUSG Transaction**
• **Usage and Procedure Logging with SAP Solution Manager**

**Define the Scope**
The CCM app allows you to specify the migration scope, that is, which custom code objects should be migrated during the conversion of your SAP ERP to SAP S/4HANA and which should not (see Figure 9).

---

**Figure 9: Scoping with the “Custom Code Migration” SAP Fiori® App**

---

© 2020 SAP SE or an SAP affiliate company. All rights reserved.
In the first step, the CCM app loads the list of all custom objects from the development system. By default, all objects are marked as “in scope for migration.” Next, it loads the system usage data (either UPL or SCMON) and applies a sophisticated logic to calculate which unused ABAP objects can be safely left behind. This logic takes into account the dependencies between the used and unused objects.

Of course, you can adjust the suggested scope manually. For instance, you can exclude objects known to become obsolete after conversion or add new developments for which no usage data yet exists in your production SAP ERP.

For step-by-step guidance, please see:
- Custom Code Analysis for SAP S/4HANA with SAP Fiori App Custom Code Migration

Prepare a Backup
Most IT organizations would insist on having a backup copy of deleted objects and a tested restore procedure before giving a green light to deletion.

If you retain a (downsized) production SAP ERP system for legal or auditing reasons, it can at the same time serve as a backup repository for the obsolete custom code. If you plan to use a data retention solution and to sunset the production system, please consult the software vendor on the available options. The ultimate option is, of course, to retain a (downsized) version of the development system.

With either option, you can use “transport of copies” to extract the required objects from the backup system and import them into the new development system. In any case, your restore procedure has to include the execution of ABAP Test Cockpit checks on the restored objects, code adaptation, and testing as mandatory steps.
Delete Objects During System Conversion
Once the scoping is complete, the CCM app generates a transport request specifying the objects subject to deletion.

During the system conversion of your development, QA, and production systems, you supply this file to Software Update Manager when it prompts you for customer transports (see Figure 10). You can also choose to use this transport request in a sandbox test cycle first to train the procedure and experience your converted system without obsolete custom code.

Figure 10: Customer Transport Prompts from Software Update Manager
AUTOMATED AND MANUAL CUSTOM CODE ADAPTATION

The following steps represent best practices for automated and manual custom code adaptation.

Plan Your Landscape
SAP generally recommends testing the conversion on a sandbox copy of the production SAP ERP system prior to performing it in the production track. Following this recommendation, many project teams also decide to use the sandbox to give the business a preview of the new system. To enable execution of custom enhancements, they have to perform custom code adaptation in the sandbox too.

In practice, the project team faces a choice between the three options outlined below (see Figure 11):

• **A long development freeze in the SAP ERP track** – While this option requires the least amount of work from IT, many businesses would probably raise an objection to a development freeze of several weeks or months.

• **Double work** – This involves redoing all custom code adaptation in the newly converted development system. This option is the simplest one but also the most labor intensive.

• **Retrofit with SAP Solution Manager** – The retrofit functionality has been your reliable companion in complex landscapes for almost a decade. With it, you neither have a development freeze nor double work. All custom code adaptation performed in the sandbox is retrofitted into the newly converted development system. This scenario is very suitable for systems with large amounts of custom code and an intense ongoing custom development. It requires SAP Solution Manager 7.2 SP8.

Removing obsolete code significantly reduces the effort for custom code adaptation. At the same time, this is certainly a first step toward a clean digital core.
**Figure 11: Options for Performing Custom Code Adaptations in a Sandbox**

**Option A – Long development freeze, hardly an option for most businesses**
- Dual maintenance for hot fixes
- Complete or partial custom code adaptation

**Option B – Double work**
- No development while conversion is in progress
- Dual maintenance for hot fixes
- Complete or partial custom code adaptation

**Option C – Retrofit**
- Retrofit of hot fixes
- Adaptation of remaining custom code
- One-time retrofit of custom code adaptation

---

**SAP Solution Manager**

---

**Transport path**
- Conversion
- System copy

---

© 2020 SAP SE or an SAP affiliate company. All rights reserved.
The latter offers many more comfort functions, such as filtering out findings in the obsolete code or displaying the coverage through quick fixes.

If you want to assess the custom code impact before the first sandbox conversion, you can use an instance of SAP Cloud Platform, ABAP environment, to execute the ATC checks remotely and analyze the result with the CCM app.

Executing ATC on a stand-alone SAP NetWeaver 7.52 system offers the least functionality (see table).

### Identify the Impact of Custom Code

Major differences in both the data model and the code base between SAP ERP and SAP S/4HANA require adaptation of custom code. All differences are maintained in the so-called simplification database. The ABAP Test Cockpit (ATC) uses the latter to identify the impact and necessary changes to custom code, including the custom code in modifications and enhancements.

SAP recommends executing ATC on an SAP S/4HANA system and using the CCM app.

### Three Options to Execute Custom Code Analysis with ABAP® Test Cockpit (ATC)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Required System</strong></td>
<td>Central ATC check system (SAP_BASIS 7.52)</td>
<td>Central ATC check system (SAP S/4HANA)</td>
<td>SAP Cloud Platform, ABAP® environment</td>
</tr>
<tr>
<td><strong>System Location</strong></td>
<td>In customer landscape</td>
<td>In customer landscape</td>
<td>Cloud</td>
</tr>
<tr>
<td><strong>Remote Connectivity</strong></td>
<td>Through RFC</td>
<td>Through RFC</td>
<td>Through RFC and the cloud connector</td>
</tr>
<tr>
<td><strong>Analysis of SAP S/4HANA® Findings</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Simplification Information in ATC Results</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Results filtered by Scope and Quick Fix Availability</strong></td>
<td>❌</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Definition of Custom Code Migration Scope Based on Usage Data</strong></td>
<td>❌</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Removal Unused Code During System Conversion with Software Update Manager</strong></td>
<td>❌</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
In either case, you can upload the check results into the SAP Readiness Check tool for SAP S/4HANA (see Figure 12). It offers a comfortable UI to browse through and classify these results.

**Figure 12: Uploading ABAP® Test Cockpit Results into SAP® Readiness Check**
The ATC can scan objects with correct ABAP syntax only. Therefore, use quick fixes to resolve the syntax errors first, and repeat the ATC check. The second iteration will most likely uncover new findings.

You may want to start with the most popular ORDER_BY fix to gain your first experience, proceed with the MATNR fix, and finally employ the fixes for data model changes, applying them sequentially for each table.

Some quick fixes offer several correction options. In that case, ask your application experts for advice.

Automated Custom Code Adaptation
With SAP S/4HANA, ABAP development tools in Eclipse enable the automated adaptation of custom code with only a few clicks through quick fixes (see Figure 13). These can resolve the most frequent findings such as ORDER_BY issues, MATNR issues, and issues related to data model changes such as KONV, BSEG, and others. SAP’s goal is to achieve a 60% to 80% automation rate in any given system.

Once your development system is converted to SAP S/4HANA, execute SAP S/4HANA–relevant ATC checks in Eclipse. Do not apply all quick fixes at once; this might overload the system and generate a log that’s hard to comprehend.

The ATC can scan objects with correct ABAP syntax only. Therefore, use quick fixes to resolve the syntax errors first, and repeat the ATC check. The second iteration will most likely uncover new findings.

You may want to start with the most popular ORDER_BY fix to gain your first experience, proceed with the MATNR fix, and finally employ the fixes for data model changes, applying them sequentially for each table.

Some quick fixes offer several correction options. In that case, ask your application experts for advice.

Automated Custom Code Adaptation
With SAP S/4HANA, ABAP development tools in Eclipse enable the automated adaptation of custom code with only a few clicks through quick fixes (see Figure 13). These can resolve the most frequent findings such as ORDER_BY issues, MATNR issues, and issues related to data model changes such as KONV, BSEG, and others. SAP’s goal is to achieve a 60% to 80% automation rate in any given system.

Once your development system is converted to SAP S/4HANA, execute SAP S/4HANA–relevant ATC checks in Eclipse. Do not apply all quick fixes at once; this might overload the system and generate a log that’s hard to comprehend.

The ATC can scan objects with correct ABAP syntax only. Therefore, use quick fixes to resolve the syntax errors first, and repeat the ATC check. The second iteration will most likely uncover new findings.

You may want to start with the most popular ORDER_BY fix to gain your first experience, proceed with the MATNR fix, and finally employ the fixes for data model changes, applying them sequentially for each table.

Some quick fixes offer several correction options. In that case, ask your application experts for advice.

Automated Custom Code Adaptation
With SAP S/4HANA, ABAP development tools in Eclipse enable the automated adaptation of custom code with only a few clicks through quick fixes (see Figure 13). These can resolve the most frequent findings such as ORDER_BY issues, MATNR issues, and issues related to data model changes such as KONV, BSEG, and others. SAP’s goal is to achieve a 60% to 80% automation rate in any given system.

Once your development system is converted to SAP S/4HANA, execute SAP S/4HANA–relevant ATC checks in Eclipse. Do not apply all quick fixes at once; this might overload the system and generate a log that’s hard to comprehend.

The ATC can scan objects with correct ABAP syntax only. Therefore, use quick fixes to resolve the syntax errors first, and repeat the ATC check. The second iteration will most likely uncover new findings.

You may want to start with the most popular ORDER_BY fix to gain your first experience, proceed with the MATNR fix, and finally employ the fixes for data model changes, applying them sequentially for each table.

Some quick fixes offer several correction options. In that case, ask your application experts for advice.

Automated Custom Code Adaptation
With SAP S/4HANA, ABAP development tools in Eclipse enable the automated adaptation of custom code with only a few clicks through quick fixes (see Figure 13). These can resolve the most frequent findings such as ORDER_BY issues, MATNR issues, and issues related to data model changes such as KONV, BSEG, and others. SAP’s goal is to achieve a 60% to 80% automation rate in any given system.

Once your development system is converted to SAP S/4HANA, execute SAP S/4HANA–relevant ATC checks in Eclipse. Do not apply all quick fixes at once; this might overload the system and generate a log that’s hard to comprehend.

The ATC can scan objects with correct ABAP syntax only. Therefore, use quick fixes to resolve the syntax errors first, and repeat the ATC check. The second iteration will most likely uncover new findings.

You may want to start with the most popular ORDER_BY fix to gain your first experience, proceed with the MATNR fix, and finally employ the fixes for data model changes, applying them sequentially for each table.

Some quick fixes offer several correction options. In that case, ask your application experts for advice.
Planning Manual Custom Code Adaptation
No conversion project escapes the debate on the effort estimation for custom code adaptation, which, of course, has to be “as accurate as possible.” The most sophisticated effort estimation models that we saw factored in the amount and variability of findings, complexity of the code, and developers’ skills – and still got it wrong.

Instead, we suggest asking your team members to study the SAP notes describing the custom code impact (the list can be found in the SAP Readiness Check tool) and letting them spend a week or two working through some of the findings in the code. After that, ask them for a range estimate. The answer you receive will be much more accurate than the result of the most sophisticated estimation model.

Finally, consider the following practical advice.

First, it is helpful to distinguish between the purely technical code corrections and the ones that require application knowledge. The former can be done by a generalist ABAP developer, while the latter requires your functional architects to look into the matter first and might lead to new development requests.

Second, development teams of any size can become much more efficient if the individual members specialize in particular code changes (that is, a set of SAP notes).

Third, use a single transport for all change requests to avoid object-locking issues.

“We believe wide ranges make us appear ignorant or incompetent. The opposite is usually the case.”
Steve McConnell, Software Estimation: Demystifying the Black Art, 2006
When it comes to SAP software modifications, the terminology seems to be a bit tricky (see Figure 14). Let’s define the terms.

A **direct modification** is a change to an SAP standard object. Some changes, such as manual correction instructions for data dictionary (DDIC) objects, are done following the guidance provided in SAP notes, while the others are performed by customers to change the standard product’s behavior.

**Indirect modifications** are an umbrella term used for:
- Clones
- Implicit enhancements at begin
- Class-method overwrite enhancements

Although the term sounds somewhat less hazardous, indirect modifications may have even worse impact on system maintainability than direct ones.

**Clones** are copies of SAP standard objects. These are often introduced to avoid making direct modifications and are used instead of original standard objects. The original objects continue receiving code corrections and additional functionality with system updates, while clones don’t. Unlike modifications and enhancements, clones are not detected during upgrades or system conversion. This is why they generally represent a risk to system stability.

**Implicit enhancements at begin** provide another possibility to overwrite standard coding, and so do **class-method overwrites**. These are listed in SPAU_ENH and should be carefully reviewed during upgrades.

Our practical experience with some heavily modified systems shows that up to 90% of modifications become dispensable after conversion to SAP S/4HANA, while up to 50% are already obsolete on SAP ERP Central Component.

Of course, this doesn’t happen by itself. You need to invest the team’s time in modification analysis.
Direct and indirect modifications increase the costs of upgrades. Up to 90% of modifications may become dispensable on SAP S/4HANA.

Here is a piece of practical advice:
- Declare “zero modifications” as one of your project’s goals.
- Treat modification analysis as a work package, not as a task, and schedule enough time for SPDD and SPAU during the sandbox conversion. Name solution architects responsible for making decisions on modification reimplementations.
- Analyze all modifications. Don’t be intimidated by the numbers; the actual modifications are usually much fewer.
- Review all SAP code clones with the clone finder tool, implicit enhancements at begin, and class-method overwrite enhancements with transaction SE84 or SPAU_ENH. Treat them as “indirect modifications” and replace them with the SAP standard code in combination with explicit enhancements.
- Use the categorization defined in the table below when working on modifications.

In some cases, the current modifications can be replaced with the in-app extensibility mechanisms in SAP S/4HANA. However, modifications can hardly be replaced with side-by-side extensions.

To track your progress, use the following KPIs:
- Number of modifications reverted
- Number of clones eliminated
- Number of required modifications

### Modification Categories

<table>
<thead>
<tr>
<th>Modification Category</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obsolete – Object identical to the SAP version</td>
<td>Revert</td>
</tr>
<tr>
<td>Unused – According to the usage statistics</td>
<td>Revert</td>
</tr>
<tr>
<td>Dispensable – Objects that become irrelevant on SAP S/4HANA® (for example, indexes) or a standard object that belongs to a deprecated application component</td>
<td>Revert</td>
</tr>
<tr>
<td>Replaceable – When you can fulfill a business requirement with standard SAP® software functionality, in-app extensibility (for example, UI adaptation), or a partner solution</td>
<td>Revert and redesign upon conversion</td>
</tr>
<tr>
<td>Required – Modification to support a critical business process</td>
<td>Document business requirements and contact SAP to get advice on those modifications</td>
</tr>
</tbody>
</table>
PERFORMANCE OPTIMIZATION OF CUSTOM CODE

The ABAP Test Cockpit offers performance checks to reveal poorly performing ABAP code. However, optimizing the entire body of custom code is usually impractical because of the associated effort. Instead, invest in performance testing and address the revealed performance issues.

Schedule a volume and performance test, and allocate time for performance optimization. Using production-like hardware for both database server and application servers is essential; our experience shows that many post-go-live issues could have been avoided through more diligent performance testing.

Experience shows that the root causes for 60% of performance issues reside in custom code. Activate the SQL monitor tool in the system from the moment it is handed over to the project. Having its data readily available will help your team resolve the reported issues in a timely manner.

SQL monitor identifies expensive SQL statements and the corresponding ABAP objects. Use ATC performance checks to analyze these objects and get hints on performance optimization.

To learn more about ABAP development tools, please visit SAP Community.
RETHINK, NOT JUST REWORK

For every custom program or report, adaptation is the most straightforward approach, yet not always the right one. Consider this:

- If you can redesign a development with the newest technological options (see the table titled “Traditional RICEFW Versus New Technologies” in “Part Three”), you will not only save the code adaptation efforts, but you may have significantly less maintenance effort in the future.

- Some custom extensions, including modifications, can be replaced with the in-app extensibility features of SAP S/4HANA; others turn out to be good candidates for side-by-side extensibility. Consider the decision matrix in “Part Four.” It will help you to make these decisions.

- Custom developments with complex code significantly increase maintenance costs. Use code complexity as a technical metric to prioritize which custom developments need a redesign. The CCM app helps you to spot the top complex executables and development packages. The most complex packages very often also experience the most frequent changes. Thus, redesigning or eliminating these bears the most saving potential (see Figure 15).

- Another candidate for redesign is “orphaned” custom functionality, that is, developments without proper documentation and/or owner. Avoid taking orphaned developments into your SAP S/4HANA system. The system conversion is the right time to reassess and decide on the options for such developments.

Figure 15: Complexity Analysis in “Custom Code Migration” SAP Fiori® App
There are a number of well-known code complexity metrics in computer science. In our experience, Halstead’s difficulty seems to be the most relevant in practice. It quantifies how difficult a program is to write or understand when applying changes.

**Combining it with the concept of h-Index** allows you to measure the complexity of entire software packages – instead of individual programs.

At the same time, we found that the code complexity correlates very well with the frequency of changes; more complex packages are changed more frequently.

Thus, combining these two analysis dimensions helps you focus on the most expensive custom developments in terms of maintenance cost (see Figure 16).

With enterprise software, both the “difficulty to write or understand” and the frequency of changes directly translate into the software maintenance cost.

For systems with a larger custom code footprint, we generally suggest starting with a review of the top 25 to 50 most complex and most frequently changed development packages (the upper right part of the diagram).

Very often the corresponding business requirements can be replaced by the new or updated functionality in SAP S/4HANA, so make sure you involve experienced SAP S/4HANA solution architects to help you identify and leverage the opportunities for complexity reduction of the solution.

---

**Figure 16: The most complex and most frequently changed custom developments are likely to drive the maintenance cost for the custom code in your system**

- Development package or executable (transaction, report, and so on)
Part Three

CUSTOM EXTENSIONS IN NEW IMPLEMENTATIONS

KEY TAKEAWAYS

• The SAP S/4HANA transition is the right moment to apply the new paradigm: clean core with decoupled extensions. And if you go for a new implementation, it’s your only chance.

• To keep your core clean, you need to change your development process, update the guidelines, and introduce control mechanisms.

• Understanding the new technological options and how they map to the traditional RICEFWs is essential.

• Your action plan for adopting SAP Cloud Platform should include setting up a technology boot camp, introducing the role of SAP Cloud Platform evangelist to lead your team’s adoption of the new technologies, and letting your team explore new technologies in a lab.

• Account for a hybrid two-tier architecture. Make extensions in a way that enables them to also work for the cloud.

• Avoid a lift-and-shift strategy for existing custom extensions. Apply the decision matrix for new business requirements the same way as for existing custom extensions in your SAP ERP before approving a lift-and-shift approach.

• Give clear guidance to the team on embedded analytics, in-app extensibility, and side-by-side extensibility as preferred options.

This part focuses on the practical aspects of managing the extensions in new implementations of SAP S/4HANA and SAP S/4HANA Cloud.
KEEP THE CORE CLEAN

A clean core paradigm requires changes in development processes and practices. Changing old habits may be hard at times. So besides updating your development guidelines, you should also establish a set of control mechanisms:

• Apply a zero-modification policy from the project’s first day and use the “Modification Overview” (transaction code SE95).

• Make sure the new code is compatible with SAP S/4HANA. Make an ABAP Test Cockpit check mandatory for transport request release and use the S4HANA_READINESS check variant.

• Use white-listed APIs only; use SAP API Business Hub\(^5\) to explore these.

• When approving development requests, check if your team has fully exploited the in-app extensibility options for UI adaptation, custom fields and tables, operational reporting, and analytics. Inserting a checklist with in-app and side-by-side extensibility options or a questionnaire into your development request template may turn out to be very useful.

Needless to say, a successful adoption of this paradigm requires your team to understand the full spectrum of the new technological options.

\(^5\) SAP API Business Hub is currently mainly focused on the SAP S/4HANA Cloud deployment option. It is gradually extended with APIs and application events for SAP S/4HANA.
UNDERSTAND THE NEW TECHNOLOGIES

SAP technologies have seen a tremendous evolution over the past decade (see Figure 17). These new technologies have enabled SAP to rearchitect one of the largest products in the history of software, and they will boost the productivity of your development team too. The table on the next page shows how the new technologies map to the traditional categories of SAP development – reports, interfaces, conversions, extensions, forms, and workflows, also known as RICEFWs.

We suggest you put the following steps into your action plan:

• Setting up a technology boot camp can be a worthwhile first step on the way to the latest technologies. Visit the openSAP platform to browse the extensive portfolio of comprehensive, open online courses centered around SAP S/4HANA and use our Learning Journey interactive guides for role- and topic-specific learning path recommendations (for example, Learning Journey guides for SAP S/4HANA developer roles). On top of it all, an SAP certification helps validate your expertise and is recognized globally.
• Next, you should appoint a team member to the role of SAP Cloud Platform evangelist. Ask them to take leadership in the adoption of new technologies and to coach the other team members on how to best apply them.
• Let your team explore these options in a lab environment to understand the types of business requests you can solve through them.
• Make the evaluation of the new technologies a mandatory step in your design and build process. Give clear guidance to the team on embedded analytics, in-app extensibility, and side-by-side extensibility as preferred options.
• Finally, update your ABAP development guidelines with the new best practices and compatibility requirements for SAP S/4HANA.

Figure 17: Custom Code Evolution at SAP

1995

SELECT * FROM Employee INTO TABLE it_empl WHERE orgunit = 4711.
LOOP AT it_empl.
  WRITE it_empl-id.
  WRITE it_empl-name.
  SELECT * FROM Addresse INTO TABLE it_addrs WHERE id = employees-id.
  LOOP AT it_addrs.
    IF it_addrs-type = 'HOMEADDR'.
      WRITE it_addrs-zipcode.
    ENDIF.
  ENDLOOP.
ENDLOOP.

2018

SELECT id, name, homeAddress.zipCode FROM Employee WHERE orgunit=4711.
A **2018 IDC research study** on the impact of training showed that properly trained implementation teams reduce software deployment times by 11%.

### Traditional RICEFW* Versus New Technologies

<table>
<thead>
<tr>
<th>Traditional RICEFWs</th>
<th>New Technology</th>
</tr>
</thead>
</table>
| **Reports (analytics)** | - Real-time analytics and KPI tiles with SAP® Smart Business cockpits, and drill-down analysis with the Analysis Path Framework service in SAP S/4HANA® embedded analytics  
- Custom analytical application with multidimensional reporting in SAP S/4HANA  
- SAP Analytics Cloud solution  
- SAP Fiori® apps |
| **Reports (automation)** | - SAP Intelligent Robotic Process Automation services |
| **Reports (applications)** | - Custom applications on SAP Cloud Platform with SAP Web IDE and SAP Cloud Platform Extension Factory, decoupled from the core through open APIs and event brokering using the SAP Cloud Platform Enterprise Messaging service  
- Custom SAP Fiori apps deployed either on SAP Cloud Platform or SAP S/4HANA (on premise) |
| **Interfaces** | - Extension of standard OData services or creation of new ones based on custom core data services (CDS) views with the in-app extensibility capabilities of SAP S/4HANA  
- SAP Cloud Platform Integration Suite  
- SAP Application Interface Framework tool (part of SAP S/4HANA)  
- Event brokering using SAP Cloud Platform Enterprise Messaging |
| **Conversion programs** | - SAP S/4HANA migration cockpit for data load |
| **Enhancements** | - Custom business logic with the in-app extensibility capabilities of SAP S/4HANA  
- SAP Cloud Platform Extension Factory, SAP Cloud Platform Enterprise Messaging |
| **Workflows** | - SAP S/4HANA flexible workflow  
- SAP Cloud Platform Workflow service |
| **Forms** | - SAP S/4HANA output management: custom forms with Adobe LiveCycle Designer with OData as data source |
| **Custom tables** | - Custom business objects with a generated UI through the in-app extensibility capabilities of SAP S/4HANA |
| **Modifications** | - You shouldn’t have to make any. In-app extensibility in SAP S/4HANA covers a wide range of business requirements for UI adaptation and business logic. |
| **User interface** | - SAPUI5 and SAP Fiori user experience |
| **Performance** | - Code pushdowns with the SAP HANA® database |

*Reports, interfaces, conversions, extensions, forms, and workflows
AVOID LIFT AND SHIFT

Reusing the custom code from today’s SAP ERP is often considered rather obvious. Many projects refer to such an approach as “lift and shift.” However, before you endorse copying the old code into the new system, we would like you to reflect on the following facts.

First, the entire body of custom code that you copy over to the new system would have to undergo the same code adaptation procedure as in a system conversion. Complex code pieces are likely to also have quality issues and may need performance optimization too. Also, you may often end up copying much more code to your system than you originally intended because of the dependencies to other development packages. Needless to say, the associated effort grows linearly with the amount of code you copy over.

Second, and more important, many of your current implementations are likely to be outdated both technically and functionally (that is, they may rely on modifications, old-style enhancement techniques, or old business logic). Therefore, you may want to establish a procedure for code reuse requests. At a minimum, such a procedure should include the following steps:

• Reconfirm the need for the current custom functionality with the business.
• Evaluate SAP and partner solutions. Investigate the SAP Store site for SAP solutions and SAP App Center or Certified Solutions Directory for partner solutions before discussing replatforming options for your custom code.
• Reevaluate if the same can be achieved with in-app extensibility.
• Reevaluate if it qualifies for a side-by-side extension on SAP Cloud Platform.

That being said, executing an “import-all” for ABAP transport requests from your current SAP ERP DEV system is probably the worst thing you can do to your new implementation.

For more details, see the section “Part Four – Key Architectural Patterns and Decision Matrix” on page 48.
STARTING WITH SAP CLOUD PLATFORM

Take the following steps when getting started with SAP Cloud Platform.

**Choose a Use Case**

Asking your team or implementation partner to showcase how the cloud can solve the most urgent challenges may seem very natural at first sight. However, there is a trap: you may end up picking a use case that’s not suitable for the cloud at all, make your implementation partner have a hard time building it, and dissatisfy your business in the end.

When talking to customers’ teams, we generally suggest starting with a use case that allows them to demonstrate the practical benefits of the cloud development. Many choose to start with an internal application to support their employees’ day-to-day tasks. Building their first application without external exposure will take some pressure off your team. At the same time, this helps everyone in the company to collect experience with cloud applications and better imagine how different business departments can benefit from the same.

**Connect Back-End Systems with SAP Cloud Platform**

Whichever use case you decide to start with, connect your SAP Cloud Platform account to your SAP S/4HANA landscape (on premise or cloud) and your identity provider. This is essential for side-by-side extensibility and will make your applications productive from day one.

Many companies have a copy of their production system. Connecting your SAP Cloud Platform development subaccount to such a system will boost your team’s productivity and help to get business buy-in when presenting prototypes or early versions on real data.

**Build for the Cloud**

With the new technological options and programming models, there is no one-to-one migration of custom applications to the cloud. You should see it as an opportunity rather than a tedious rework. When making the decision to redesign, be aware of the new efficiencies brought to you through new technologies, new application programming models, and new programming language features.

The first change is about the UI. The application’s user interface will require a redesign with the cloud UI technologies. Take this opportunity to rethink the user experience. Apply design thinking methods, and ensure that you continuously collect users’ feedback. If, at some stage, you realize that a new UI actually requires a complete application redesign, then the time has come for such a change.

The second big change is about data access. The application’s access to the ERP data will have to switch from SQL to API calls. Start by deciding where to keep the application-specific data: in SAP S/4HANA in the form of custom business objects, or in the SAP HANA instance in your SAP Cloud Platform account.

With the first option, you should use the in-app extensibility capabilities of SAP S/4HANA to define custom business objects and generate OData services for remote access. With the second one, a best practice is to define a separate database schema for each cloud application to simplify application deployment between DEV, QA, and PROD subaccounts. You can then access the data through custom-built OData services or directly from your Java or ABAP code.
Apply DevOps Practices
These tips can help you put DevOps to work in your development organization:

• Ask a technology expert from your team to study the DevOps methodology, take ownership of the pipeline including any extra tool you may want to plug in, and coach the development team(s).

• Choose and configure the code quality checks and security scans to match corresponding standards in your organization. Do not rely on default settings.

• Adhere to “build once and fail fast” principles as well as the mandatory participation and continuation of the scrum ceremonies.

• Constantly improve the development process and automation tools. However, better does not mean more control. DevOps is not about strict rules and governance but about the efficiency of the development teams. Keep this in mind when conducting process reviews.

• Consider the DevOps metrics listed in the following table. These will help you to have a clear picture of your efficiency, code equality, and applications usage. Most pipeline tools offer APIs to retrieve the corresponding data points.

• Clearly, each organization needs to decide for itself on the target values for these KPIs. Consider the table on the next page titled “Aspect of Software Delivery Performance” when measuring the performance of your team.

DevOps Metrics

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead time for changes</td>
<td>Average time it takes from idea, feature request, or backlog item to a successful deployment in the productive system environment</td>
</tr>
<tr>
<td>Deployment frequency</td>
<td>Number of deployments for an application in a given time frame</td>
</tr>
<tr>
<td>Change failure rate</td>
<td>Percentage of changes that led to service degradation or issues</td>
</tr>
<tr>
<td>Mean time to recover (MTTR; also known as time to restore service)</td>
<td>How long it takes to recover from a failed deployment in production</td>
</tr>
<tr>
<td>Backlog burn-down</td>
<td>Number of backlog items submitted versus implemented per time frame</td>
</tr>
<tr>
<td>Number of bugs</td>
<td>Number of confirmed software issues</td>
</tr>
<tr>
<td>Number of security vulnerabilities</td>
<td>Number of security vulnerabilities detected through code checks and number of security vulnerabilities detected in production</td>
</tr>
<tr>
<td>Number of incidents</td>
<td>Number of incidents raised</td>
</tr>
<tr>
<td>Number of users</td>
<td>Number of users accessing the application in a given time frame</td>
</tr>
</tbody>
</table>

According to Accelerate: State of DevOps 2018: Strategies for a New Economy, a report by DevOps Research & Assessment (DORA), the highest performers manage to bring new code changes and code corrections into production within an hour, effectively delivering multiple code deployments a day.

6. Notice the extended definition of the metric.
### Aspect of Software Delivery Performance

<table>
<thead>
<tr>
<th></th>
<th>Elite&lt;sup&gt;a&lt;/sup&gt;</th>
<th>High</th>
<th>Medium</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deployment frequency</strong>&lt;br&gt;For the primary application or service you work on, how often does your organization deploy code?</td>
<td>On-demand (multiple deploys per day)</td>
<td>Between once per hour and once per day</td>
<td>Between once per week and once per month</td>
<td>Between once per week and once per month</td>
</tr>
<tr>
<td><strong>Lead time for changes</strong>&lt;br&gt;For the primary application or service you work on, what is your lead time for changes (that is, how long does it take to go from code commit to code successfully running in production)?</td>
<td>Less than one hour</td>
<td>Between one day and one week</td>
<td>Between one week and one month</td>
<td>Between one month and six months&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td><strong>Time to restore service</strong>&lt;br&gt;For the primary application or service you work on, how long does it generally take to restore service when a service incident occurs (for example, unplanned outage or service impairment)?</td>
<td>Less than one hour</td>
<td>Less than one day</td>
<td>Less than one day</td>
<td>Between one week and one month</td>
</tr>
<tr>
<td><strong>Change failure rate</strong>&lt;br&gt;For the primary application or service you work on, what percentage of changes results either in degraded service or subsequently requires remediation (for example, leads to service impairment, service outage, requires a hotfix, rollback, fix forward, or patch)?</td>
<td>0%–15%</td>
<td>0%–15%</td>
<td>0%–15%</td>
<td>46%–60%</td>
</tr>
</tbody>
</table>

Medians reported distributions are not normal. All differences are significantly different based on Tukey’s post hoc analysis except where otherwise noted.

<sup>a</sup>The elite performance group is a subset of the high performance group.

<sup>b</sup>Means are not significantly different based on Tukey’s post hoc analysis; medians exhibit differences because of underlying distribution. Typical low performers have a lead time for changes between one month and six months, and typical medium performers have a lead time for changes between one week and one month, however, tests for significant differences show that overall, these two groups are not statistically different when including all group members’ variance in behavior.

These four patterns capture the essentials of the architecture for the corresponding class of extensions as they respond to these two questions:

- What technologies does your extension utilize?
- How do you deploy the UI and the business logic, and where is the data of your extension persisted?

In practice, your applications may combine these patterns. For instance, the application could follow the “cloud application” pattern and, at the same time, extend the business objects in SAP S/4HANA with “in-app extensibility.”
PATTERN #1: CLOUD APPLICATION

With this pattern, you create a new cloud application on SAP Cloud Platform and integrate it to SAP and third-party, on-premise, and cloud products based on standard and custom APIs (see Figure 18).

Figure 18: Cloud Application Pattern
The following table provides an overview of the cloud application pattern.

<table>
<thead>
<tr>
<th>When to use the cloud application pattern</th>
<th>Traits</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Build business process extensions integrated with SAP S/4HANA® and other SAP® products through standard or custom APIs</td>
<td>• Cloud development is automated with the DevOps pipeline.</td>
</tr>
<tr>
<td>• Target end users outside the back end or intranet (B2C, B2B, B2E)</td>
<td>• Business logic is implemented on SAP Cloud Platform using Java or Node.js (with SAP Cloud SDK and SAP Cloud Application Programming Model), ABAP® programming language (using ABAP RESTful Programming Model), or workflow and business rules engines.</td>
</tr>
<tr>
<td>• Complement the data model of the main business solution with IoT data or data from open data sources (for example, social media)</td>
<td>• Application-specific data is stored on SAP Cloud Platform.</td>
</tr>
<tr>
<td>• Decouple your custom extensions from the back-end lifecycle, availability, and load (24x7, holiday peak sales scenarios, year-end close reporting)</td>
<td>• Integration to SAP solutions such as SAP S/4HANA, SAP S/4HANA Cloud, SAP SuccessFactors® solutions, and others is established through standard APIs or custom APIs built with in-app extensibility.</td>
</tr>
<tr>
<td>• Create custom workflows and reactive (event-based) process extensions</td>
<td>• Possibly, data is replicated into SAP Cloud Platform for consumption by custom application and/or analytics with SAP Analytics Cloud.</td>
</tr>
<tr>
<td>• Provide data analytics across multiple back-end systems with possible data consolidation in the SAP Analytics Cloud solution</td>
<td>• Single identity provider allows for single sign-on and principal propagation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>• This pattern provides a common platform and extension mechanism for all SAP cloud and on-premise products.</td>
<td>• The absence of standard APIs and/or standard business objects to build a custom API may require building custom APIs with classic custom development. This work-around introduces dependency on the back end’s lifecycle.</td>
</tr>
<tr>
<td>• It enables true decoupling of custom extensions from SAP code. The clear list of consumed APIs simplifies risk impact analysis and test scoping during back-end upgrades.</td>
<td>• This pattern supports eventual data consistency. Custom data stored in the cloud cannot be changed together with core data in the back end.</td>
</tr>
<tr>
<td>• The application’s release cycle is independent from the back-end release cycles. Daily or even more frequent application deployments are possible.</td>
<td>• The decoupled lifecycle and lean dependencies allow for quick proof of concept (PoC) and minimal viable product (MVP) development.</td>
</tr>
<tr>
<td>• The decoupled lifecycle and lean dependencies allow for quick proof of concept (PoC) and minimal viable product (MVP) development.</td>
<td>• Isolated development environments (separate “space” per development team) and API access control allow for collaboration with external development teams without providing access to back-end systems.</td>
</tr>
</tbody>
</table>

Have your voice heard. Join the Customer Influence program to submit your requests for new product APIs.
PATTERN #2: HYBRID APPLICATION

With this pattern, you improve UX by exposing your application’s user interface through SAP Cloud Platform while the application logic and data reside in your SAP S/4HANA on premise or on another back-end system (see Figure 19).

Figure 19: Hybrid Application Pattern
The following table provides an overview of the hybrid application pattern.

<table>
<thead>
<tr>
<th>When to use the hybrid application pattern</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Provide mobile and external access to existing applications built in the ABAP® programming language on SAP S/4HANA® or SAP® Business Suite applications (B2C, B2B, B2E)</td>
<td></td>
</tr>
<tr>
<td>• Improve UX by redesigning the UI for existing applications (for example, simplifying data-entry screens, dropping screens that are not required, auto-filling fields, and enabling speech-to-text, translation, and localization functionality)</td>
<td></td>
</tr>
<tr>
<td>• Automate data entry using a service or device</td>
<td></td>
</tr>
<tr>
<td>• Provide data analytics for a single or several back-end systems</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Traits</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Application logic is developed on the back end using classical development or ABAP CDS views. The data is exposed to the UI through custom REST (OData) services.</td>
<td></td>
</tr>
<tr>
<td>• Single identity provider allows for single sign-on and principal propagation.</td>
<td></td>
</tr>
<tr>
<td>• Separate (cloud-only) identity provider allows external users to access the application. Back-end connectivity is then based on a technical user.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benefits</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Allows you to provide mobile and external access with a modern UI to the existing custom applications with some moderate effort</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Restrictions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Release cycle is dependent on the back-end release cycle.</td>
<td></td>
</tr>
<tr>
<td>• Existing custom applications in the back end might not be optimized for the cloud-based UI, leading to a high number of back-end calls and extensive data transfer. A redesign of certain application logic on the back end might be required to optimize the performance.</td>
<td></td>
</tr>
</tbody>
</table>
PATTERN #3: IN-APP EXTENSIBILITY

With this pattern, you consume the in-app extensibility capabilities of SAP S/4HANA to create custom extensions with key user tools and embedded analytics (see Figure 20).

Figure 20: In-App Extensibility Pattern
The following table provides an overview of the in-app extensibility pattern.

<table>
<thead>
<tr>
<th>When to use the in-app extensibility pattern</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• UI adaptation of standard SAP Fiori® apps: add/hide/relocate fields, add custom fields</td>
<td></td>
</tr>
<tr>
<td>• Change in business logic with custom logic (BAdI implementations)</td>
<td></td>
</tr>
<tr>
<td>• Creation of custom business objects with automatic generation of tables, maintenance UI, and OData services</td>
<td></td>
</tr>
<tr>
<td>• Creation of custom views and custom APIs for data consumption from external applications</td>
<td></td>
</tr>
<tr>
<td>• Custom analytical applications using built-in analytical capabilities and frameworks, such as SAP® Smart Business cockpits</td>
<td></td>
</tr>
<tr>
<td>• Custom print and e-mail forms</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Traits</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Managed enhancements in SAP S/4HANA® and SAP S/4HANA Cloud</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benefits</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Most of the changes require no ABAP® development and can be done by key users.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Restrictions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• UI adaptation works with SAP Fiori apps only. Applications based on the SAP GUI interface for HTML (Web GUI) and the Web Dynpro development environment have limited extensibility support.</td>
<td></td>
</tr>
</tbody>
</table>

Have your voice heard. Join the **Customer Influence** program and submit your requests for **new extensibility options**.
PATTERN #4: CLASSICAL EXTENSION

With this pattern, you reuse the enhancement techniques and custom development capabilities from your SAP ERP Central Component system (see Figure 21).

The following table provides an overview of the classical extension pattern.

| When to use the classical extension pattern | • Reuse custom development from your SAP® ERP application (SAP ERP Central Component 6.x) as is |
| Traits | • Use of the classical UI technologies (SAP GUI interface, Web Dynpro development environment)  
• Tight coupling of custom code to standard SAP code |
| Benefits | • Existing application can be reused in SAP S/4HANA®. Custom code adaptation will still be required (see “Part Two”). |
| Restrictions | • Release cycle is dependent on the back-end release cycle. |
**DECISION MATRIX**

The purpose of the decision matrix in the table below is to help your architects to choose the best-fitting architecture pattern when evaluating new business requirements or deciding on the future of the existing custom developments in SAP ERP.

<table>
<thead>
<tr>
<th>Requirements Toward Custom Functionality</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Users and UX</strong></td>
<td>Cloud application</td>
</tr>
<tr>
<td>Involve consumers of the corporate products and services (B2C) (for example, service orders, master data self-services, catalogs, Web shops, mobile access)</td>
<td>+</td>
</tr>
<tr>
<td>Involve business partners (B2B) to enable direct collaboration (for example, order review and approval, service or good receipt, quality control, delivery checkpoints)</td>
<td>+</td>
</tr>
<tr>
<td>Involve employees (B2E) who otherwise have no access to the business solution (for example, outsourced workers, leased workers, mobile workers)</td>
<td>+</td>
</tr>
<tr>
<td>Adapt existing UIs based on the SAP Fiori® UX – Add, hide, move, or regroup fields on screen, add custom fields, change label texts</td>
<td>+</td>
</tr>
<tr>
<td>Improve UX by redesigning the UI for existing applications (for example, simplifying data-entry screens, dropping screens that are not required, auto-filling fields, and enabling speech-to-text, translation, and localization functionality)</td>
<td>(+)(^{b})</td>
</tr>
<tr>
<td>Open-source components and freestyle UI (non-SAPUI5/SAP Fiori)</td>
<td>+</td>
</tr>
<tr>
<td>Mobile native capabilities (for example, access to microphone, camera, GEO location, and so on)</td>
<td>+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Data</strong></th>
<th>Cloud application</th>
<th>Hybrid application</th>
<th>In-App extensibility</th>
<th>Classical extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stand-alone application based on own data model with occasional consumption of standard data in SAP S/4HANA(^{a})</td>
<td>+</td>
<td></td>
<td>(+)(^{c})</td>
<td></td>
</tr>
<tr>
<td>Analytical application consuming standard and custom data residing in SAP S/4HANA</td>
<td>(+)(^{d})</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analytical application consuming data distributed across multiple SAP(^{a}) and non-SAP systems (for example, data lake)</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transactional data consistency – Custom data is changed in a single database transaction with core data in the back end.</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Features</strong></th>
<th>Cloud application</th>
<th>Hybrid application</th>
<th>In-App extensibility</th>
<th>Classical extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agility and independence on the back-end lifecycle</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reactive (event-based) process extensions and custom workflows</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use of SAP and third-party cloud services (for example, machine learning solutions from SAP, SAP Localization Hub services, tax services, Google Maps, and so on)</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application with unpredictable or largely varying usage and resource consumption (scalability and elasticity)</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^{a}\)UI adaptation for SAP GUI for HTML (Web GUI) and Web Dynpro applications require modifications and classic enhancement techniques.

\(^{b}\)Data entry automation, for instance, might require application logic to be implemented on SAP Cloud Platform.

\(^{c}\)Simple use cases can be easily implemented with the help of in-app extensibility.

\(^{d}\)Use the SAP Analytics Cloud solution for data analysis while building a custom API for data provisioning.
ADDITIONAL CONSIDERATIONS

Here are a few thoughts to conclude this section.

The architectural design decision may be influenced by compliance and legal aspects, such as legal restrictions on storing the data in the cloud. Involve your security team and clarify the boundaries beforehand.

However, the two aspects below should neither influence your choice of the pattern nor affect your strategy for adoption of the new technologies.

The first is the devotion to SAP GUI. We acknowledge that there will be teams who want to retain SAP GUI because they are more productive with it on tasks that require massive manual data entry or data changes. If you try offering them a 1:1 SAP Fiori—like substitute, you will likely experience very little acceptance. Instead, a new app will have to offer new and unique features that allow for massive time savings. Thus, a wiser strategy would be to offer both options in parallel for some time. Allow users to explore and to compare – and may the better one win.

The same holds true for your development team. In practice, we have seen teams so used to classic ABAP that they would hardly consider any other option. At the same time, we have never seen a team that would create a solid software architecture by following the “design to skills” approach. Therefore, we can only reiterate the importance of upskilling your architects and development teams on the new technologies. The faster your team gains new skills and becomes proficient with the modern software architectures and development tools, the sooner you will be able to provide your business with the speed and agility it needs to thrive in the modern global economy.
Conclusion

We hope that this guide helps you understand the new concepts for extending SAP solutions and how to leverage the new technologies in your SAP S/4HANA projects.

While we have highlighted the pivotal elements of SAP Cloud Platform, we certainly haven’t covered all of them. For those who want to learn more, we can recommend these main resources:

- Developing Resilient Apps on SAP Cloud Platform
- Tutorial Navigator
- Learning Journeys for Developers
- SAP Fiori Deployment Options and System Landscape Recommendations (October 2019)
- SAP Cloud Platform Portal and SAP Enterprise Portal Road Map

We highly appreciate your feedback. Please don’t hesitate to share it with us at s4move@sap.com. In the future, we intend to publish more guides on the practical aspects of the transition to SAP S/4HANA.
Acknowledgments

We would like to thank Ralf Peters from DSAG for his contribution, which helped us to combine SAP’s point of view as a software vendor with the collective experience of customers to focus on the right subjects. We would also like to thank Christian Freytag and Bjoern Braemer for initiating the creation of this handbook, and Aleksandr Kirov and Peter Datsichin for making it happen. Finally, we would like to thank all of our coworkers at SAP for their tireless support, especially:

- Martin Bachmann
- Thomas Bieser
- Olga Dolinskaja
- Thomas Fiedler
- Sabrina Hartmann
- Benjamin Heilbrunn
- Henning Heitkoetter
- Marco Matha
- Niklas Meinert
- Rui Nogueira
- Patrick Schmidt
- Thomas Schneider
- Paolo Erhardt Sfilio
- Nancy Teich
- Andrej Titov
- Florian Wahl
- Wolfgang Weiss
- Felix Wente
- Bjoern Woppmann
Glossary

**ABAP** – Advanced Business Application Programming, SAP’s programming language

**ALE** – Application Link Enabling, an SAP technology that allows the exchange of data messages, thus enabling data consistency across loosely coupled applications

**API** – Application programming interface, a technique that allows external access to the application’s data, business objects, and their methods. In SAP applications, these are BAPIs, RFC-enabled function modules, and OData services, for example.

**ATC** – ABAP Test Cockpit tool for static custom code analysis (transaction code: ATC)

**BAdI** – Business add-in; SAP object-oriented enhancement technique available as of SAP R/3 4.6c. Business add-ins are predefined by SAP within the SAP ERP application and allow customers to enhance the business logic by adding custom code in BAdI implementations. BAdI custom implementations are triggered in runtime by standard application logic.

**BAPI** – Business Application Programming Interface is an SAP API technology that allows users to read and update data. It may be called from within the SAP system or remotely using RFC protocol and ALE/IDoc technology. Several BAPI calls can be wrapped into a single LUW (transaction).

**BSEG** – A database table in SAP ERP that stores segment data in an accounting document

**BTE** – Business Transaction Event, an SAP enhancement type developed for financial accounting

**CDS** – Core data services is an infrastructure used to create semantically rich data models in the SAP HANA database with native support for PFCG authorizations, associations, annotations, and extensions. CDS views are created in ABAP or directly in SAP HANA while executed on SAP HANA in runtime. CDS is widely used for code pushdown.

**CSS3** – Cascading Style Sheets language used to describe the presentation of HTML documents

**DDIC** – Data dictionary in SAP R/3. It stores definitions of objects, such as tables, data elements, domains, and search helps.

**DevOps** – A set of practices spanning and building a closed loop over all stages of the software lifecycle, from development to operations

**ERP** – Enterprise resource planning software

**h-Index** – A ranking function used as an aggregation function in the code complexity analysis

**HTML5** – Fifth generation of the software solution stack that defines the properties and behaviors of Web page content by implementing a markup-based pattern to it (Source: Wikipedia)

**IDoc** – Intermediate document, an SAP standard for electronic messages to exchange information between SAP and non-SAP systems

**Include** – An ABAP object containing source code that can be included with other programs

**IoT** – Internet of Things

**jQuery** – JavaScript library

**KONV** – A database table in SAP ERP that stores conditions data

**LESS** – Dynamic preprocessor style sheet language that can be compiled into Cascading Style Sheets and run on the client side or server side (Source: Wikipedia)
LUW – Logical unit of work, a concept that allows users to write multiple changes from several dialog steps into the database in a single database transaction

MATNR – Material number data element. This data element has 18 characters in standard SAP ERP, while it has been extended to 40 characters in SAP S/4HANA. The custom code referencing material number has to be inspected and, possibly, adjusted upon system conversion.

ML – Machine learning applications and services

OData – The Open Data Protocol allowing the creation and consumption of query-able and interoperable RESTful APIs in a simple and standard way (Source: Wikipedia)

ORDER_BY – A reference to the issue with SQL statements in custom code that do not have explicit sorting instructions, while the following application logic relies on the data set being sorted. Most relational databases return the data sorted by used database index, while SAP HANA returns unsorted data sets due to parallel data retrieval. The ORDER BY clause or SORT ABAP statement has to be added to all affected SQL statements upon database migration to SAP HANA.

PFCG – Transaction code for the administration of authorizations, profiles, and roles

Quick fix – Functionality of ABAP development tools in Eclipse that allows for automated custom code refactoring

RESTful – Representational state transfer (REST) is an architectural style for creating Web services as a set of stateless operations. Such Web services are called RESTful services.

Retrofit – An SAP technology to transfer changes to customizing or DDIC and coding objects from system to system

RFC – Remote function call

RICEFW – Reports, interfaces, conversions, extensions, forms, and workflows

SAPUI5 – SAP’s main UI framework that is built on HTML5, CSS3, JavaScript, LESS, and jQuery standards and enhanced to match the needs of enterprise application development

SCMON – Transaction code of the ABAP Call Monitor tool, a successor of UPL, that tracks and stores the names of coding objects executed in the system in runtime

SE38 – Transaction code for “ABAP Editor”

SE95 – Transaction code for “Modifications Overview”

SMOD/CMOD – Transaction codes used to maintain “Customer Exits,” which is an SAP enhancement technique

SPAU/SPDD – Transaction codes of tools used to adjust customer modifications during system upgrade or system conversion

SPAU_ENH – Transaction code of a tool used to adjust customer enhancements during system upgrade or system conversion

SQL – Structured Query Language, a language used for managing data in relational databases

SUSG – Transaction code of a tool that provides aggregation functionality for the usage data collected by the ABAP Call Monitor

TCO – Total cost of ownership

UI – User interface

UPL – Usage and Procedure Logging, an SAP technology that tracks and stores the names of coding objects executed in the system in runtime

User exit – An SAP enhancement technique

UX – User experience

VOFM routines – An SAP enhancement technique

Web Dynpro – An SAP UI development environment

Web GUI – SAP technology that allows running SAP GUI and Web Dynpro applications in a browser