SAP S/4HANA

# Custom Extensions in SAP S/4HANA® Implementations

## A Practical Guide for Senior IT Leadership

May 2021

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

# Table of Contents


3 Highlights May 2021 Update

4 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

13 Key User (In-App) Extensibility

15 Developer (On-Stack) Extensibility

16 Side-by-Side Extensibility on SAP BTP

22 DevOps

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

45 Avoid Lift and Shift

46 Starting with SAP BTP

51 Part Four – Key Architectural Patterns and Decision Matrix

52 Pattern #1: Cloud Application

54 Pattern #2: Hybrid Application

56 Pattern #3: Key User (In-App) Extension

58 Pattern #4: Developer (On-Stack) Extension

60 Pattern #5: Classical Extension

61 Decision Matrix

63 Additional Considerations

64 Conclusion

65 Acknowledgments

66 Glossary




© 2021 SAP SE or an SAP affiliate company. All rights reserved.

# Highlights
# MAY 2021
# UPDATE

- Our cloud platform (formerly called SAP Cloud Platform) is one of the core pillars powering SAP BTP, rather than a stand-alone offering. As such, it no longer requires its own brand. All services under the former SAP Cloud Platform brand have been renamed accordingly.
- In 2021, we introduced a new developer (on-stack) extensibility option. Read more on that in "Part One" and explore a new architectural pattern ("Pattern #4: Developer (On-Stack) Extension") in "Part Four."
- The continuous delivery pipeline of SAP® Cloud SDK has been merged into the general purpose pipeline of project "Piper." The latter is also offered as the SAP Continuous Integration and Delivery service on SAP Business Technology Platform (SAP BTP). The corresponding section has been moved from "Part One" to "Part Three."
- We have added new courses to the openSAP platform on custom extensions.

**Check here for the online version of the latest updates.**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

# Foreword by DSAG

Standing out from competitors while keeping up technical standardization is going to be a great challenge for many, especially with the increasing importance of the cloud.

The DSAG therefore welcomes this guide, which offers valuable support and practical tips for SAP customers who deal with enhancements in their SAP S/4HANA® projects. The document supports developers, system administrators, and project teams in implementing such distinctive features without "leaving the path of the standard", thus remaining or becoming cloud-ready.

The guide explains the most important concepts for creating custom extensions in SAP S/4HANA with the help of SAP® Business Technology Platform (SAP BTP). The combination of SAP BTP and the industry cloud will help customers to achieve the above goals, which also includes transferring today's modifications into the "new world". The possibilities shown here go far beyond SAP S/4HANA. SAP BTP is the glue in SAP's future architecture as it gives you the freedom of being different and, thus, also freedom to innovate.

As DSAG, we appreciate this guide, especially in combination with the guide for IT managers, **"Mapping Your Journey to SAP S/4HANA - A Practical Guide for Senior IT Leadership"**, the SAP's handbook for planning a successful migration as well as further DSAG and SAP guides.

**Thomas Henzler**
DSAG Board Member Licenses & Maintenance.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

# Foreword by Bjoern Braemer

**"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, 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**
SVP & Program Management Executive,
Business and Data Transformation Solutions,
SAP One Movement
SAP SE



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

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 the transaction code 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.

**Figure 1: A Time Journey**

© 2021 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 this: allow customers to extend their SAP S/4HANA software while making the eventual updates non-events.
- Key user (in-app) extensibility is a powerful tool that allows you to implement the typical extension in SAP S/4HANA – mostly without a single line of code.
- SAP® Business Technology Platform (SAP BTP) 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. Key user (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 BTP.

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

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 key user (in-app) extensibility and side-by-side extensibility, and cloud development (see Figure 2). 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 2: Key Concepts for Custom Extensions in SAP S/4HANA®**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## CONSUMER-GRADE UX

The digitalization of society has profoundly changed the way users think of enterprise applications. Younger generations starting their career in your company expect 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" – 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 capabilities include SAP S/4HANA embedded analytics, situation handling, intelligent scenario lifecycle management, and digital assistants, for example.

SAP has been continuously developing new SAP Fiori apps while visually harmonizing traditional SAP GUI interface transactions and ABAP applications created using Web Dynpro with the SAP Fiori visual theme. For an excellent introduction to the concepts of SAP Fiori, review the **SAP Fiori Design Guidelines**. More information on SAPUI5 is available in the demo kit for **SAPUI5 SDK**.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 released APIs only
- Leveraging the key user (in-app) extensibility of SAP S/4HANA to its full extent
- Employing the capabilities and services offered by SAP BTP to build larger extension applications
- Utilizing SAP Integration Suite

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.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 key user, or 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 3). As a matter of fact, both types of extensibility are mostly used in combination; therefore, it is almost never an either/or decision.

The key user (in-app) extensibility within SAP S/4HANA and the side-by-side extensibility on SAP BTP cover a lot of extensibility use cases. Still, some business requirements lead to extensions that are too complex to be built with the former and too tightly connected to core ERP data and business processes to be built outside of SAP S/4HANA. Until now, organizations had to rely on traditional development practices to address such requirements.

Starting in 2021,[1] SAP plans to release a new "developer (on-stack) extensibility" option that will allow you to build custom applications and enhance standard processes in your SAP S/4HANA applications and SAP S/4HANA Cloud tenants. This option, of course, will follow the clean core paradigm.

We explain these types of extensibility in the next sections.

Let us add a remark on the classical 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.

1. This is the current state of planning and may be changed by SAP at any time without notice.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

**Figure 3: Extensibility Framework of SAP S/4HANA®**

*Not available in SAP S/4HANA Cloud

User interface layer  Application layer  Database layer



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## KEY USER (IN-APP) EXTENSIBILITY

The idea of key user extensibility, also known as 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 key user (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 key user (in-app) extensibility patterns (see Figure 3):

- **Adopt existing SAP Fiori apps** – Add, hide, move, or regroup fields on-screen, add custom fields, and change label texts. Add, hide, move, or resize cards in overview pages. Hide or move buttons in toolbars. Set preferred links in smart link dialogs.
- **Add application 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 can be made 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 released 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 on SAP BTP) 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 BTP, and deploy it to your SAP S/4HANA system or SAP S/4HANA Cloud tenant.

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

Key user (in-app) extensibility and side-by-side extensibility are **mostly used in combination**; it is almost never an either/or decision.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

Much like customizing, these adaptations are made in the development (DEV) system and transported to production with 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 key user enhancements made.

Please visit the **SAP Extensibility Explorer** site or read the blog titled "**The Key User Extensibility Tools of SAP S/4HANA**" to learn more about the extensibility patterns and the technology behind them.

Key user (in-app) extensibility is a powerful tool that allows you to implement **typical extensions** in SAP S/4HANA – mostly without a single line of code.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## DEVELOPER (ON-STACK) EXTENSIBILITY

Starting in 2021,[2] SAP plans to offer an additional extensibility option as part of SAP S/4HANA and SAP S/4HANA Cloud to complement current key user (in-app) extensibility capabilities. The planned developer (on-stack) extensibility option will empower ABAP developers to extend standard SAP processes with custom extensions that require tighter integration with the SAP S/4HANA back end while keeping the digital core isolated from the custom code. This will help minimize the efforts of regression testing during SAP S/4HANA upgrades.

Technically, this will allow you to develop custom extensions on the same ABAP platform that the SAP S/4HANA instance or SAP S/4HANA Cloud tenant uses. Clear separation between custom and standard code is achieved by adopting the ABAP RESTful application programming model based on CDS views and optimized ABAP, along with released DDIC objects and local APIs. You can use the developer (on-stack) extensibility option to extend standard processes, create custom OData services to be consumed by locally deployed SAP Fiori apps, or build custom APIs for integration.

Similar to SAP BTP for the ABAP environment, the planned developer (on-stack) extensibility option will allow you to reuse some of your application logic and ABAP code from your SAP ERP system. And just like on SAP BTP, classical ABAP capabilities, such as SAP GUI transactions and programs, modifications, and classical enhancement techniques, will be unavailable.

Your main tools will be ABAP development tools in Eclipse for development in ABAP and SAP Business Application Studio for the SAP Fiori UX.

**2.** This is the current state of planning and may be changed by SAP at any time without notice.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## SIDE-BY-SIDE EXTENSIBILITY ON SAP BTP

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

- It 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 **comprehensive security** from the UI to the back end.
- It enables smooth **connectivity** to SAP solutions and easier discovery and consumption of APIs and events across applications.
- It is integrated into the lifecycle management of **hybrid landscapes**.

SAP BTP 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 4).

### Connect and Communicate

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

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

Establishing such connections has been greatly simplified through the extensibility services to connect systems to SAP BTP. The number of manual steps (such as exchanging certificates or entering URLs) has been massively reduced. It now takes just a few clicks.[3]

**3.** Enabled for SAP S/4HANA Cloud, the SAP Customer Experience portfolio, and SAP SuccessFactors solutions; other solutions are on the road map.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## Figure 4: Side-by-Side Extensibility with SAP® Business Technology Platform

*This is the current state of planning and may be changed by SAP at any time without notice.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

Once your SAP solutions have been connected to SAP BTP, 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 5).

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 Customer Experience portfolio, and the SAP Sales Cloud and SAP Service Cloud solutions natively support application events. These events can be published through the SAP Event Mesh capability, enabling applications to 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 available APIs and events and allows users to try them out directly in their browser.

**Figure 5: Available APIs Exposed Using Extensibility Services to Connect to SAP® Business Technology Platform**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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, SAP Fiori elements, workflow, business rules)
- 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 BTP 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 the ABAP RESTful[4] application programming model).

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

The choice of the programming language 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.

4. 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.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## SAP® Cloud Application Programming Model Versus ABAP® RESTful Application Programming Model

| Common to Both Models | ABAP® RESTful Application Programming Model | SAP® Cloud Application Programming Model |
|---|---|---|
| • RESTful OData services for communication between the front end (UI) and the application on SAP® Business Technology Platform (SAP BTP)<br>• 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<br>• 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<br>• Built-in extensibility capabilities, enabling users to extend both SAP Cloud Application Programming Model and ABAP® RESTful application programming model in a similar way as with the key user (in-app) extensibility of SAP S/4HANA® (see above) | • Supports the ABAP programming language<br>• Provides Git-enabled lifecycle management<br>• Offers a service consumption model for easy remote OData service calls<br>• Enables the possibility to reuse selected custom code in the cloud with SAP BTP, ABAP environment, while rebuilding the UI and back-end access<br>• Leverages modern ABAP development tools in Eclipse for back-end development on premise or in the cloud | • Supports Java or JavaScript (node.js)<br>• Enables applications originally written as single-tenant applications to be turned into multitenant ones through configuration<br>• Leverages event-based communication using the SAP Event Mesh capability<br>• Wraps the REST service calls to the underlying back-end system to Java or JavaScript functions using SAP Cloud SDK |



© 2021 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 API Management capability of SAP Integration Suite provides you with the 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 Business Technology Platform capabilities
- SAP Extensibility Explorer site
- SAP Discovery Center site
- "Introduction to SAP Extension Suite" course on the openSAP platform
- "Create and Deliver Cloud-Native SAP S/4HANA Extensions" course on openSAP
- "Extending SAP S/4HANA with SAP Business Technology Platform" course on openSAP
- "SAP Business Technology Platform Essentials" course on openSAP
- SAP Business Technology Platform trial
- *Customize Your SAP S/4HANA System with SAP Cloud Platform* book
- "Building Apps with the ABAP RESTful Application Programming Model" course on openSAP



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

# DEVOPS

DevOps is a set of practices spanning and building a closed loop over all stages of the software lifecycle, from development to operations. It subsumes the ideas of agile development, continuous integration (CI), continuous delivery (CD), continuous deployment, and continuous operations and adoption (see Figure 6).

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 the following:

- 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 Accelerate: State of DevOps 2019 Report by DevOps Research and Assessment LLC (DORA) to the readers looking for actionable guidance on how to improve their software delivery performance.

**Figure 6: 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.

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

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**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 SAP Fiori app "Custom Code Migration" in SAP S/4HANA and the Software Update Manager tool.

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

Technically, the procedure includes the following steps:

- Collecting usage data – This is done through activating usage and procedure logging (UPL) or the ABAP call monitor (SCMON for short) in the production instance of your SAP ERP.
- Defining the scope – Based on the usage data, the Custom Code Migration app proposes a set of objects that can be safely deleted. You can refine this proposal manually in your sandbox for SAP S/4HANA.
- 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.[5]

**Figure 8: Deletion of Unused Custom Code Integrated into Conversion**

5. 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.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

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 guide

## Define the Scope

The Custom Code Migration 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 SAP Fiori® App "Custom Code Migration"**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

In the first step, the Custom Code Migration app loads the list of custom objects from the development system. By default, 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, read the blog "**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 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 must include the execution of ABAP test cockpit checks on the restored objects, code adaptation, and testing as mandatory steps.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## Delete Objects During System Conversion

Once the scoping is complete, the Custom Code Migration 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**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 the conversion 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 a 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 suitable for systems with large amounts of custom code and an intense ongoing custom development. It requires SAP Solution Manager 7.2 support package 8.

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.

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## Figure 11: Options for Performing Custom Code Adaptations in a Sandbox

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 uses the latter to identify the impact and necessary changes to custom code, including the custom code in modifications and enhancements.

SAP recommends executing the ABAP test cockpit on an SAP S/4HANA system and using the Custom Code Migration app.

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 BTP for the ABAP environment to execute checks remotely in the ABAP test cockpit and analyze the result with the Custom Code Migration app.

Executing the ABAP test cockpit on a stand-alone SAP NetWeaver 7.52 system offers the least functionality (see the table below).

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

| | Remote ABAP® Test Cockpit with SAP® GUI | SAP Fiori® app "Custom Code Migration" | SAP Fiori app "Custom Code Migration" on SAP BTP |
|---|---|---|---|
| Required System | Central ABAP test cockpit check system (SAP_BASIS 7.52) | Central ABAP test cockpit check system (SAP S/4HANA) | SAP BTP, ABAP® environment |
| System Location | In customer landscape | In customer landscape | Cloud |
| Remote Connectivity | Through RFC | Through RFC | Through RFC and the cloud connector |
| Analysis of SAP S/4HANA® Findings | ✓ | ✓ | ✓ |
| Simplification Information in ABAP Test Cockpit Results | ✓ | ✓ | ✓ |
| Results Filtered by Scope and Quick Fix Availability | X | ✓ | ✓ |
| Definition of Custom Code Migration Scope Based on Usage Data | X | ✓ | ✓ |
| Removal of Unused Code During System Conversion with Software Update Manager | X | ✓ | ✓ |



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

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

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 ABAP test cockpit 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 ABAP test cockpit can scan objects with correct ABAP syntax only. Therefore, use quick fixes to resolve the syntax errors first, and repeat the ABAP test cockpit 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.

**Figure 13: Custom Code Adaptation with Quick Fixes**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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**

**Quick fixes** automate custom code adaptation. They are available with ABAP development tools in Eclipse as part of your SAP S/4HANA license.

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## REVIEW OF MODIFICATIONS, CLONES, AND IMPLICIT ENHANCEMENTS

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.

Of course, this doesn't happen by itself. You need to invest the team's time in modification analysis.

**Figure 14: Types of SAP® Software Modifications**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

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 reimplementation.
- 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 key user (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

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



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

# 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.

The SQL monitor tool identifies expensive SQL statements and the corresponding ABAP objects. Use the performance checks in the ABAP test cockpit to analyze these objects and get hints on performance optimization.

To learn more about ABAP development tools, please visit SAP Community.

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 key user (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 Custom Code Migration 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 SAP Fiori® app "Custom Code Migration"**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

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 to reduce the complexity of the solution.

**Figure 16: Factors Driving the Maintenance Cost of Custom Code**

● = Development package or executable (transaction, report, and so on)



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

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 BTP should include setting up a technology boot camp, introducing the role of SAP BTP 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, key user (in-app) extensibility, and side-by-side extensibility as preferred options. For on-premise development, make use of the "restricted ABAP" programming language, a special version of ABAP initially created for SAP BTP. Restricted ABAP now lets you split your developments into cloud-ready parts and access classes on SAP S/4HANA.

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

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 released APIs only; use SAP API Business Hub[6] to explore these.
- When approving development requests, check if your team has fully exploited the key user (in-app) extensibility options for UI adaptation, custom fields and tables, operational reporting, and analytics. Inserting a checklist with key user (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.

SAP's rationale behind the **clean digital core paradigm** is this: allow customers to extend their SAP S/4HANA software while making the eventual updates non-events.

6. 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.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 page 44 shows how the new technologies map to the traditional categories of SAP software 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 SAP Learning Journey interactive guides for role- and topic-specific learning path recommendations (for example, SAP 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 BTP 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, key user (in-app) extensibility, and side-by-side extensibility as preferred options.
- For on-premise development, make use of the restricted ABAP language, which is a special version of ABAP initially created for SAP BTP in the ABAP environment. This allows you to split your developments into cloud-ready parts and access classes. The latter use standard ABAP to call unreleased function modules and make direct access to standard tables. Access classes also help you bridge the time until necessary APIs are released in SAP S/4HANA.
- Finally, update your ABAP development guidelines with the new best practices and compatibility requirements for SAP S/4HANA.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## Figure 17: Custom Code Evolution at SAP

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## Traditional RICEFW* Versus New Technologies

| Traditional RICEFWs | New Technology |
|---|---|
| **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<br>• Custom analytical application with multidimensional reporting in SAP S/4HANA<br>• Other SAP Fiori® apps, such as overview pages and analytical list pages<br>• SAP Analytics Cloud solution |
| **Reports (automation)** | • SAP Intelligent Robotic Process Automation services |
| **Reports (applications)** | • Custom applications on SAP Business Technology Platform (SAP BTP) with SAP Web IDE and SAP Extension Suite, decoupled from the core through open APIs and event brokering using the SAP Event Mesh capability<br>• Custom SAP Fiori apps deployed either on SAP BTP, SAP S/4HANA, or SAP S/4HANA Cloud |
| **Interfaces** | • Extension of standard OData services or creation of new ones based on custom core data services (CDS) views with SAP S/4HANA key user (in-app) extensibility<br>• SAP Integration Suite<br>• SAP Application Interface Framework tool (part of SAP S/4HANA)<br>• Event brokering using SAP Event Mesh |
| **Conversions** | • SAP S/4HANA migration cockpit to load data |
| **Enhancements** | • Custom business logic with SAP S/4HANA key user (in-app) extensibility<br>• SAP Extension Suite, SAP Event Mesh |
| **Workflows** | • SAP S/4HANA flexible workflow<br>• SAP Workflow Management service |
| **Forms** | • SAP S/4HANA output management: custom forms with Adobe LiveCycle Designer with OData as data source |
| **Custom tables** | • Custom business objects with generated UI with SAP S/4HANA key user (in-app) extensibility |
| **Modifications** | • You shouldn't have to make any. Key user (in-app) extensibility in SAP S/4HANA covers a wide range of business requirements for UI adaptation and business logic. |
| **User interfaces** | • SAP Fiori and SAPUI5 |
| **Performance** | • SAP HANA® code pushdown |

*Reports, interfaces, conversions, extensions, forms, and workflows

A recent 2020 IDC research study[7] on the impact of training showed that well-trained implementation teams and administrators can save up to 15% in deployment time.

7. Cushing Anderson, "Game Changer: The Transformative Impact of Training," IDC white paper sponsored by SAP, October 2020.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 the **Certified Solutions Directory** for partner solutions before discussing replatforming options for your custom code.
- Reevaluate if the same can be achieved with key user (in-app) extensibility.
- Reevaluate if it qualifies for a side-by-side extension on SAP BTP.

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 51**.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## STARTING WITH SAP BTP

Take the following steps when getting started with SAP BTP.

### 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 BTP

Whichever use case you decide to start with, connect your SAP BTP account to your SAP S/4HANA or SAP S/4HANA Cloud landscape 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 BTP 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 BTP account.

With the first option, you should use the key user (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.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 titled "Aspect of Software Delivery Performance" when measuring the performance of your team.

## The Continuous Integration and Delivery Pipeline

The technical infrastructure for executing DevOps is usually composed of several tools. These tools are connected sequentially and build a "pipeline" for the code to flow from development to production. Such a pipeline helps validate every change to your application by running it through a set of automated stages. Continuous integration helps ensure that every change checked in by a developer leads to a consistent state of the whole application by running builds, unit tests, and security scans. Continuous delivery automates the application's deployment into QA, and once an (optional) approval is received, into production.

Setting up your own continuous delivery pipeline can be labor intensive and prone to errors. SAP offers various options to help you create continuous delivery pipelines depending on your requirements and expertise:

- The SAP Continuous Integration and Delivery service together with the SAP Cloud Transport Management service on SAP BTP let you configure and run predefined continuous integration and delivery pipelines that test, build, and deploy your code changes. They support typical extension development scenarios, using SAP Cloud Application Programming Model and SAPUI5 or SAP Fiori.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

- Project "Piper"[8] provides a prebuilt continuous delivery stack based on an open-source automation server called Jenkins. This stack can be instantiated on any machine equipped with Docker, or in a Kubernetes cluster. "Piper" is fully flexible, extendable, and open source and enables you to reuse your existing continuous integration and continuous delivery tool stack (see Figure 18).

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.

**Change Management in Hybrid Landscapes**

There are several solutions that help you accelerate the deployment of your hybrid application.

A Git-enabled change and transport system (gCTS) can be used for selected projects. This allows creation of a separate pipeline for ABAP development.

If your development is led with ABAP in the back end while the UI presents on SAP BTP (for example, see "Pattern #2: Hybrid Application" in "Part Four"), use the enhanced change and transport system (CTS+) in SAP NetWeaver. It helps ensure simultaneous update of ABAP code and UI parts; on the SAP BTP side, it is limited to artifacts that can be wrapped into a multitarget application (MTA) archive (such as SAPUI5 apps, Java apps, and portal content).

**Figure 18: General Purpose Pipeline**

8. Project "Piper" is an open-source project led by SAP. The formerly offered continuous delivery pipeline of SAP Cloud SDK has been migrated into the general purpose pipeline of project "Piper." Migration is described here.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

If you have established a change management workflow using the change request management (ChaRM) tool in SAP Solution Manager, use the continuous integration pipeline up to QA and then hand over control to the ChaRM tool that integrates with transport management (SAP Cloud Transport Management or CTS+) for deployment to the production account. The SAP Cloud Transport Management service will also allow you to handle SAP BTP objects that can't be packed into an MTA package.

Otherwise, the deployment process in the cloud (SAP BTP pipeline) and on premise (ABAP transport management system) are kept independent. Make sure you push back-end changes prior to UI changes while keeping the former backwards compatible.

## DevOps Metrics

| Metrics | |
|---|---|
| **Lead time for changes**[9] | Average time it takes from idea, feature request, or backlog item to a successful deployment in the productive system environment |
| **Deployment frequency** | Number of deployments for an application in a given time frame |
| **Change failure rate** | Percentage of changes that led to service degradation or issues |
| **Mean time to recover (MTTR; also known as time to restore service)** | How long it takes to recover from a failed deployment in production |
| **Backlog burn-down** | Number of backlog items submitted versus implemented per time frame |
| **Number of bugs** | Number of confirmed software issues |
| **Number of security vulnerabilities** | Number of security vulnerabilities detected through code checks and number of security vulnerabilities detected in production |
| **Number of incidents** | Number of incidents raised |
| **Number of users** | Number of users accessing the application in a given time frame |

According to "Accelerate: State of DevOps 2018: Strategies for a New Economy," a report by DevOps Research and 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.

9. Notice the extended definition of the metric.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## Aspect of Software Delivery Performance

| | Elite[a] | High | Medium | Low |
|---|---|---|---|---|
| **Deployment frequency**<br>For the primary application or service you work on, how often does your organization deploy code? | On demand (multiple deploys per day) | Between once per hour and once per day | Between once per week and once per month | Between once per week and once per month |
| **Lead time for changes**<br>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)? | Less than one hour | Between one day and one week | Between one week and one month | Between one month and six months[b] |
| **Time to restore service**<br>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)? | Less than one hour | Less than one day | Less than one day | Between one week and one month |
| **Change failure rate**<br>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 or outage or requires a hotfix, rollback, fix forward, or patch)? | 0%–15% | 0%–15% | 0%–15% | 46%–60% |

Medians reported distributions are not normal.
All differences are significantly different based on Tukey's post hoc analysis except where otherwise noted.
[a] The elite performance group is a subset of the high performance group.
[b] 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.

Source: "Accelerate: State of DevOps 2018: Strategies for a New Economy," DevOps Research and Assessment (DORA), 2018.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

Part Four

# KEY ARCHITECTURAL PATTERNS AND DECISION MATRIX

This section describes the architectural patterns you are most likely to lean on when building custom extensions for SAP S/4HANA. No matter if you are assessing a new business requirement or rethinking an existing custom application, these five patterns constitute the main options:

- Pattern #1: Cloud application
- Pattern #2: Hybrid application
- Pattern #3: Key user (in-app) extension
- Pattern #4: Developer (on-stack) extension
- Pattern #5: Classical extension

These five 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 Pattern #1 and, at the same time, extend the business objects in SAP S/4HANA with Pattern #3.

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## PATTERN #1: CLOUD APPLICATION

With this pattern, you create a new cloud application on SAP BTP and integrate it to SAP and third-party on-premise, cloud, and hybrid products based on standard and custom APIs (see Figure 19).

**Figure 19: Cloud Application Pattern**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

The following table provides an overview of the cloud application pattern.

| | |
|---|---|
| **When to use the cloud application pattern** | • Build business process extensions integrated with SAP S/4HANA® and other SAP® products through standard or custom APIs.<br>• Target end users outside the back end or intranet (business to consumer, business to business, business to employee).<br>• Complement the data model of the main business solution with IoT data or data from open data sources (for example, social media).<br>• Decouple your custom extensions from the back-end lifecycle, availability, and load (24x7, holiday peak sales scenarios, year-end close reporting).<br>• Create custom workflows and reactive (event-based) process extensions.<br>• Provide data analytics across multiple back-end systems with possible data consolidation in the SAP Analytics Cloud solution. |
| **Traits** | • Cloud development is automated with the DevOps pipeline.<br>• Business logic is implemented on SAP Business Technology Platform (SAP BTP) using Java or Node.js (with SAP Cloud SDK and SAP Cloud Application Programming Model), ABAP® programming language (with the ABAP RESTful application programming model), or workflow and business rules engines.<br>• Application-specific data is stored on SAP BTP.<br>• 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 key user (in-app) extensibility.<br>• Possibly, data is replicated into SAP BTP for consumption by custom application and/or analytics with SAP Analytics Cloud.<br>• Single identity provider allows for single sign-on and principal propagation.<br>• Separate (cloud-only) identity provider allows external users to access the application. |
| **Benefits** | • This pattern provides a common platform and extension mechanism for all cloud, on-premise, and hybrid solutions from SAP.<br>• 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.<br>• The application's release cycle is independent from the back-end release cycles. Daily or even more frequent application deployments are possible.<br>• The decoupled lifecycle and lean dependencies allow for quick proof of concept and minimal viable product development.<br>• 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. |
| **Restrictions** | • The absence of standard APIs and/or standard business objects to build a custom API may require building custom APIs using embedded extensions or classic custom development. The former is preferable since the latter introduces dependency on the back end's lifecycle.<br>• This pattern supports eventual data consistency. Custom data stored in the cloud cannot be changed together with core data in the back end. |

Have your voice heard. Join the **Customer Influence** program to submit your requests for **new product APIs** into the corresponding LoB-specific or cross-application session.

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## PATTERN #2: HYBRID APPLICATION

With this pattern, you improve the UX by exposing your application's UI through SAP BTP while the application logic and data reside in SAP S/4HANA or any other back-end SAP or third-party system (see Figure 20).

**Figure 20: Hybrid Application Pattern**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

The following table provides an overview of the hybrid application pattern.

| | |
|---|---|
| **When to use the hybrid application pattern** | • Provide mobile and external access to existing applications built in the ABAP® programming language on SAP S/4HANA® or SAP® Business Suite applications (business to consumer, business to business, business to employee).<br>• 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).<br>• Automate data entry using a service or device.<br>• Provide data analytics for a single or several back-end systems. |
| **Traits** | • Application logic is developed on the back end using classical development or the ABAP® RESTful application programming model. The data is exposed to the UI through custom REST (OData) services.<br>• Single identity provider allows for single sign-on and principal propagation.<br>• Separate (cloud-only) identity provider allows external users to access the application. |
| **Benefits** | • Allows you to provide mobile and external access with a modern UI to the existing custom applications with some moderate effort |
| **Restrictions** | • Release cycle is dependent on the back-end release cycle.<br>• 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. |



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## PATTERN #3: KEY USER (IN-APP) EXTENSION

With this pattern, you consume the key user (in-app) extensibility capabilities of SAP S/4HANA to create custom extensions with key user tools and embedded analytics (see Figure 21).

**Figure 21: Key User (In-App) Extension Pattern**

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

The following table provides an overview of the key user (in-app) extension pattern.

| | |
|---|---|
| **When to use the key user (in-app) extensibility pattern** | • UI adaptation of standard SAP Fiori® apps: add/hide/relocate fields, add custom fields, change label texts. Add/hide/relocate or resize cards in overview pages. Hide/relocate buttons in toolbars. Set preferred links in smart link dialogs<br>• Change in business logic with custom logic (implementation of business add-ins)<br>• Creation of custom business objects with automatic generation of tables, maintenance UI, and OData services<br>• Creation of custom views and custom APIs for data consumption from external applications<br>• Custom analytical applications using built-in analytical capabilities and frameworks, such as SAP® Smart Business cockpits<br>• Custom print and e-mail forms |
| **Traits** | • This pattern provides common managed enhancements and embedded analytical capabilities for SAP S/4HANA® and SAP S/4HANA Cloud. |
| **Benefits** | • Most of the changes require no ABAP® development and can be done by key users. |
| **Restrictions** | • 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. |

Have your voice heard. Join the **Customer Influence** program and submit your requests for **new extensibility options** into the corresponding LoB-specific or cross-application session.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## PATTERN #4: DEVELOPER (ON-STACK) EXTENSION

With this pattern, you build custom applications and enhancements of the SAP standard processes in your SAP S/4HANA applications and SAP S/4HANA Cloud tenants using an optimized ABAP language and directly accessible local APIs (see Figure 22). In comparison to the side-by-side extension pattern, this approach is built for extension use cases that require tighter integration with the SAP S/4HANA back end.

SAP plans to make this extensibility option available for SAP S/4HANA Cloud in 2021.[10] This extensibility option is already partially available in SAP S/4HANA; we plan to further develop it with new releases.

**Figure 22: Developer (On-Stack) Extension Pattern**

10. This is the current state of planning and may be changed by SAP at any time without notice.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

The following table provides an overview of the developer (on-stack) extension pattern.

| | |
|---|---|
| **When to use the developer (on-stack) extension pattern** | • Build large extensions and custom applications with tight integration with SAP S/4HANA® Cloud.<br>• Build custom applications in SAP S/4HANA that follow the clean core paradigm and keep the digital core isolated from the custom code.<br>• Transform custom application logic from your SAP® ERP application to SAP S/4HANA or SAP S/4HANA Cloud.<br>• Reuse custom extensions in hybrid two-tier landscapes[11] between SAP S/4HANA Cloud and SAP S/4HANA installations. |
| **Traits** | • Use of modern UI technologies (such as SAP Fiori® apps)<br>• Based on the ABAP® RESTful application programming model<br>• Custom extension running on the same ABAP platform stack and the same database as SAP S/4HANA or SAP S/4HANA Cloud software<br>• Extension that is clearly separated from standard code by using optimized ABAP programming language and released APIs<br>• Utilization of SAP S/4HANA Cloud extension points, local APIs, public APIs, and SAP Business Technology Platform (SAP BTP) services |
| **Benefits** | • Additional flexibility for extensions within SAP S/4HANA Cloud<br>• Direct access to all released objects and APIs of the back end of SAP S/4HANA Cloud<br>• Increased compatibility between extensions built on SAP S/4HANA Cloud, SAP S/4HANA, and SAP BTP, ABAP environment |
| **Restrictions** | • Cloud-optimized ABAP language scope to help ensure cloud readiness and low effort for applying SAP updates<br>• Enforced use of released APIs, core data services, and data elements and domains to help ensure stable operations |

**THE HYBRID TWO-TIER ARCHITECTURE**

Read more in our free white paper, "Mapping Your Journey to SAP S/4HANA – A Practical Guide for Senior IT Leadership."

11. For the hybrid two-tier architecture, please refer to the "Cloud Options" chapter in the "Mapping Your Journey to SAP S/4HANA – A Practical Guide for Senior IT Leadership" document.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## PATTERN #5: CLASSICAL EXTENSION

With this pattern, you reuse the enhancement techniques and custom development capabilities from your SAP ERP application (see Figure 23).

**Figure 23: Classical Extension Pattern**

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 6.x) "as is" on SAP S/4HANA® or SAP S/4HANA Cloud |
| **Traits** | • Use of the classical UI technologies (SAP GUI interface, Web Dynpro development environment) or reuse of existing custom SAPUI5 or SAP Fiori® apps deployed on SAP ERP<br>• Tight coupling of custom code to standard SAP code |
| **Benefits** | • Existing custom application can be reused in SAP S/4HANA; custom code adaptation will still be required (see "Part Two").<br>• Classic enhancement technologies (user or customer exits, enhancement points or spots, and more) as well as modifications provide flexible options for application tuning. However, modifications and some enhancement types have a negative impact on the lifecycle and are not recommended. |
| **Restrictions** | • Release cycle is dependent on the back-end release cycle.<br>• No modification of SAP standard code is allowed, and some extension techniques are not allowed in SAP S/4HANA Cloud. |



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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.

| | Requirements Toward Custom Functionality | Pattern: Cloud application | Hybrid application | Key user (in-app) extension | Developer (on-stack) extension | Classical extension |
|---|---|---|---|---|---|---|
| Users and UX | Involve consumers of the corporate products and services (business to consumer) (for example, service orders, master data self-services, catalogs, Web shops, mobile access) | + | + | | | |
| | Involve business partners (business to business) to enable direct collaboration (for example, order review and approval, service or good receipt, quality control, delivery checkpoints) | + | + | | | |
| | Involve employees (business to employee) who otherwise have no access to the business solution (for example, outsourced workers, leased workers, mobile workers) | + | + | | | |
| | Adapt existing UIs based on the SAP Fiori® user experience (for example, add, hide, move, or regroup fields on-screen, add custom fields, change label texts) | | | + | | (+)[a] |
| | Improve the user experience by redesigning the UI for existing applications (for example, simplifying data-entry screens, dropping screens not required, auto-filling fields, and enabling speech-to-text, translation, and localization functionality) | + | (+)[b] | | | |
| | Use open-source components and a freestyle UI (non-SAPUI5 or SAP Fiori) | + | + | | | |
| | Enable mobile native capabilities (for example, access to a microphone, camera, GEO location) | + | + | | | |

[a] UI adaptation for SAP GUI for HTML (Web GUI) and applications created using Web Dynpro require modifications and classic enhancement techniques. This is not applicable for SAP S/4HANA Cloud.
[b] Data entry automation, for instance, might require application logic to be implemented on SAP Business Technology Platform.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

| | Requirements Toward Custom Functionality | Pattern: Cloud application | Hybrid application | Key user (in-app) extension | Developer (on-stack) extension | Classical extension |
|---|---|---|---|---|---|---|
| Data | An extension to a standard business process or an application with an extensive use of data in SAP S/4HANA® | | + | | + | (+)[e] |
| | Stand-alone application based on own data model with occasional consumption of standard data in SAP S/4HANA | + | | (+)[c] | | |
| | Analytical application consuming standard and custom data residing in SAP S/4HANA | | (+)[d] | + | | |
| | Analytical application consuming data distributed across multiple SAP® and third-party systems (for example, data lake) | + | (+)[d] | | | |
| | Transactional data consistency – Custom data changed in a single database transaction with core data in the back end | | + | + | + | (+)[e] |
| Features | Agility and independence on the back-end lifecycle | + | | | | |
| | Reactive (event-based) process extensions and custom workflows | + | | | | |
| | Use of SAP and third-party cloud services (for example, machine learning solutions from SAP, SAP Localization Hub services, tax services, Google Maps) | + | | | | |
| | Application with unpredictable or largely varying usage and resource consumption (scalability and elasticity) | + | | | | |

[c] Simple use cases can be easily implemented with the help of key user (in-app) extensibility.
[d] Use the SAP Analytics Cloud solution to analyze data while building a custom API for data provisioning.
[e] Not applicable for SAP S/4HANA Cloud



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## 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 adopting 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.



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

# 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 BTP, 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
- SAP Learning Journey guides for developers
- "SAP Fiori Deployment Options and System Landscape Recommendations (April 2021)"
- SAP S/4HANA, ABAP, and SAPUI5 road maps

For a wider range of topics concerning the SAP S/4HANA transition, read our white paper titled "Mapping Your Journey to SAP S/4HANA – A Practical Guide for Senior IT Leadership."

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.

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

# 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
Lukas Bretschneider
Carine Tchoutouo Djomo
Olga Dolinskaja
Thomas Fielder
Sabrina Hartmann
Benjamin Heilbrunn
Henning Heitkoetter
Martin Huvar
Marco Matha
Niklas Meinert
Oliver Nocon
Rui Nogueira
Patrick Schmidt
Thomas Schneider
Paolo Erhardt Sfilio
Christoph Szymanski
Nancy Teich
Andrej Titov
Florian Wahl
Wolfgang Weiss
Felix Wente
Bjoern Woppmann
Boris Zarske

© 2021 SAP SE or an SAP affiliate company. All rights reserved.

# 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.

**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).

**BDEF** – A business definition that describes the behavior of a business object or application. The BDEF is defined using the Behavior Definition Language (BDL).

**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)



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

**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 applications created using Web Dynpro in a browser



© 2021 SAP SE or an SAP affiliate company. All rights reserved.

## Join the SAP S/4HANA Movement

Time to make your move:
**www.sap.com/jointhemovement**

Follow us

**www.sap.com/contactsap**

**Studio SAP | 66913enUS (21/10)**

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

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platforms, directions, and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See www.sap.com/copyright for additional trademark information and notices.

THE BEST RUN SAP®