Darstellung und Vergleich mehrerer Möglichkeiten zur Umsetzung eines sequentiellen HR-Prozesses im RESTful API-Umfeld

Projektarbeit 1

im Rahmen der Prüfung zum

Bachelor of Science (B.Sc.)

des Studienganges Wirtschaftsinformatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Tom Wolfrum

- Sperrvermerk -

Abgabedatum:	4. September 2023
Bearbeitungszeitraum:	05.06.2023 - 03.09.2023
Kurs:	WWI22B5
Ausbildungsfirma:	SAP SE Dietmar-Hopp-Allee 16 69190 Walldorf, Deutschland
Betreuer der Ausbildungsfirma:	Steven Rösinger
Gutachter der Dualen Hochschule:	Paul Peitz

Sperrvermerk

Die nachfolgende Arbeit enthält vertrauliche Daten der:

SAP SE
Dietmar-Hopp-Allee 16
69190 Walldorf, Deutschland

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen ausserhalb des Prüfungs- und Evaluationsverfahrens zugänglich gemacht werden, sofern keine anders lautende Genehmigung des Dualen Partners vorliegt.

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Projektarbeit 1 mit dem Thema:

> *Darstellung und Vergleich mehrerer Möglichkeiten zur Umsetzung eines sequentiellen HR-Prozesses im RESTful API-Umfeld*

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, den 16. August 2023

_____

Wolfrum, Tom

Geschlechtsneutrale Formulierung

In dieser Arbeit wird aus Gründen der besseren Lesbarkeit das generische Maskulinum verwendet.
Weibliche und anderweitige Geschlechteridentitäten werden dabei ausdrücklich mitgemeint, soweit es für die Aussage erforderlich ist.

Inhaltsverzeichnis

Abkürzungsverzeichnis

SaaS	Software-as-a-Service
AIS	Application Innovation Services
HCM	Human Capital Management
API	Application Programming Interface
REST	Representational State Transfer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
RAP	Restful Application Programming Model
ABAP	Advanced Business Application Programming
CDS	Core Data Service
BO	Business Object
UI	User Interface (Benutzeroberfläche)
TCO	Total Cost of Ownership (Gesamtkosten für Entwicklung und Betrieb)
BW	Business Workflow
BE	Business Events
bgPF	background Processing Framework
LUW	Logical unit of work
bgRFC	background Remote Function Call
BTP	Business Technology Platform
BD	Business Object Behavior Definition
BSD	Business Service Definition

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Unternehmensprofil und Anwendungsbezug

SAP SE ist ein börsennotierter Softwarekonzern mit Sitz in Walldorf. Das Hauptgeschäft des 1972 gegründeten Unternehmens ist die Entwicklung von Unternehmenssoftware zur Abwicklung von Geschäftsprozessen. Heute erwirtschaften 105.000 Mitarbeiter in 157 Ländern einen Umsatz von ca. 30 Mrd. €. Erfolgreich wurde das Unternehmen mit dem Verkauf von ERP Standardsoftware. In den letzten Jahren stand die Transformation des gesamten Produkt-Portfolios in Richtung Cloud-Services als Abo-Modell im Fokus der Unternehmensstrategie.[1]

Die Abteilung AIS HCM ist Teil des Unternehmensbereichs Product Engineering und zuständig für den Development-Support und Neuentwicklungen der SAP Personallösung HCM. Diese deckt Prozesse rund um das Personalwesen ab. Zudem stellt die Abteilung mehrere SAP Fiori Apps als Self-Service für Mitarbeiter bereit.

1.2. Motivation und Problemstellung

Die von der Abteilung betriebenen Fiori Apps, die schon im Zusammenhang der Einleitung angesprochen wurden, sind auf Basis des Frameworks SAP UI5 Freestyle für ein älteres Produkt - SAP ECC - entwickelt worden. Durch die strategische Entscheidung HCM als Bestandteil von S/4 zu integrieren, finden die S/4-Design-Guidelines darauf Anwendung, die z. B. Oberflächen-Design oder zu verwendende Technologien, festlegen. Fiori Apps müssen dadurch die Technologie Fiori Elements verwenden. Existierende Apps werden auf Basis der älteren Technologie in S/4 weiterbetrieben und neu entwickelte Apps müssen Fiori Elements verwenden.

Diese Situation sorgt für ein Problem in Geschäftsprozessen, die über solche Apps abgebildet werden sollen. Das Framework Fiori Elements generiert das gesamte Frontend der Anwendung selbstständig. Das erleichtert auf der einen Seite die Entwicklung der Apps, auf der anderen Seite kann dadurch keine eigene Programmlogik

[1] Vgl. SAP 2023a.

mehr im Frontend eingebaut werden. Zudem bietet das Programmiermodell RAP, das in Fiori Elements für eine konsistente Durchführung der Datenbankoperationen zuständig ist, nur einen transaktionalen Kontext für das Ausführen von eigener Logik. Jedoch ist es in bestimmten Geschäftsprozessen nötig, asynchron in einem weiteren transaktionalen Kontext noch Programmcode ausführen zu können. Da RAP das modellseitig nicht zulässt, soll in der vorliegenden Arbeit nun untersucht werden, wie sich solche asynchronen Prozesse, trotz den eben dargelegten Einschränkungen im S/4 Umfeld mit den neueren Technologien umsetzen lassen.

1.3. Aufbau und Ziel der Arbeit

Zunächst wird die Arbeit klar abgegrenzt, sowie die wissenschaftlichen Methoden kurz genannt. Der theoretische Teil befasst sich mit der Darstellung der theoretischen Grundlagen einer RESTful API und deren Umsetzung im RESTful Application Programming Model der SAP, sowie die Erläuterung der Technologie Fiori Elements. Es folgt die Vorstellung und Gegenüberstellung von drei Technologien Business Workflows, Business Events und dem Background Processing Framework zur Lösung der in der Problemstellung adressierten Frage. Anhand von Kriterien wie Stärken und Schwächen, Effizienz und Robustheit werden die Technologien verglichen und in einer Entscheidungsmatrix dargestellt. Die Matrix soll als Orientierungshilfe dienen, welche Technologie sich am besten zur Abbildung asynchroner Prozesse in einem RESTful API-Umfeld eignet. Abschließend werden die Ergebnisse zusammengefasst und kritisch reflektiert, Handlungsempfehlungen gegeben und ein Ausblick auf zukünftige Entwicklungen präsentiert.

1.4. Abgrenzung

Der Zweck der vorliegenden Arbeit ist es, die drei vorgestellten Technologien vergleichend zu bewerten und je nach Anwendungsfall eine Handlungsempfehlung im Bezug auf eine sich anbietende Technologie zu geben. Über diese drei Technologien hinaus werden keine anderen Möglichkeiten asynchrone Prozesse abzubilden, wie z. B. im Cloud Application Programming Model (CAP) behandelt. Außerdem findet aufgrund des beschränkten Umfangs der Arbeit lediglich ein Vergleich der Technologien statt und keine direkte Implementierung dieser in einem konkreten Anwendungsfall. Hier-

für sei auf die offizielle Dokumentation der SAP mit Showcases für die respektiven Technologien verwiesen. [2]

1.5. Methodisches Vorgehen

In dieser Arbeit wurde die Methode der Experteninterviews angewendet, um spezifische SAP-Technologien zu untersuchen und in Bezug auf vordefinierte Vergleichskriterien zu bewerten. Experteninterviews ermöglichen eine umfassende Erkenntnisgewinnung und den Zugang zu praxisrelevanten Informationen, die oft nicht ausreichend in der Fachliteratur abgebildet sind. Bei der Durchführung wurde eine Mischform aus strukturierten und unstrukturierten Interviews gewählt, welche klare Fragestellungen mit Flexibilität bei der Beantwortung verbindet. Dieser Ansatz erlaubte eine Anpassung an individuelle Anwendungsfälle und trug zur Identifizierung von Best Practices und relevanten Herausforderungen in diesem spezifischen SAP-Umfeld bei.

Die Vergleichskriterien werden bei den einzelnen Ansätzen durch Experteninterviews bewertet und dann die betrachteten Ansätze anhand dieser Kriterien gegenübergestellt. So kommt dann die Entscheidungsmatrix zustande, welche Technologie sich bei welchen Anforderungen und Rahmenbedingungen anbietet.

[2] Vgl. https://github.com/SAP-samples/abap-platform-fiori-feature-showcase

2. Theoretische Grundlagen

2.1. RESTful Application Programming Interface

Eine API ist eine Schnittstelle, über die verschiedene Softwareanwendungen miteinander kommunizieren können. Die API definiert die Methoden, Protokolle und Tools, die für den Zugriff auf die Funktionen und Daten einer Softwareanwendung verwendet werden können. Somit standardisiert eine API die Kommunikation verschiedener Anwendungen und ermöglicht den Zugriff auf bereitgestellte Daten, ohne dass die zugreifende Anwendung die interne Logik oder Implementierung der anderen Anwendung kennen muss.[1]

Eine RESTful-API ist eine spezielle Schnittstelle, die den Designkonventionen nach REST folgt. Diese sind in der nachfolgenden Grafik dargestellt und werden im Folgenden erklärt.

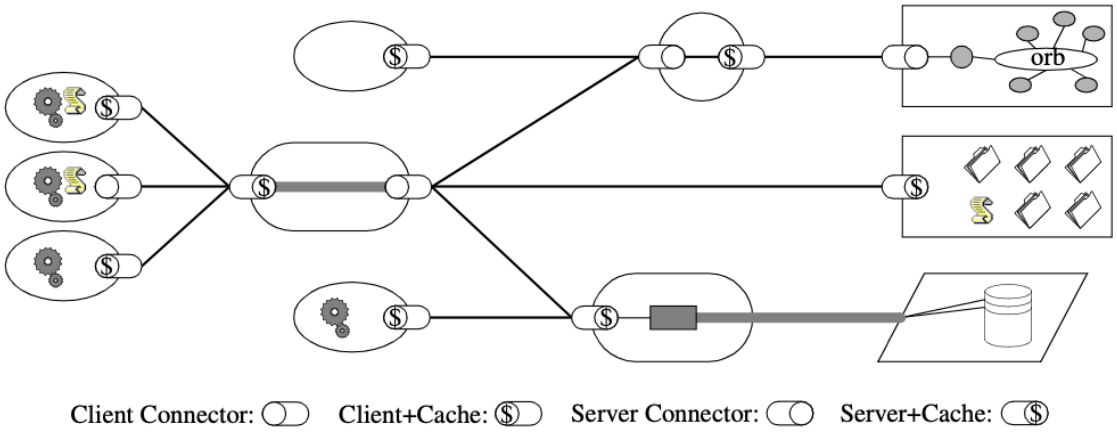


Abbildung 1.: REST Designprinzipien. Abgerufen von Fielding 2000 am 05.07.2023.

Client-Server-Architektur

Das erste Prinzip ist die Client-Server-Architektur. Das bedeutet, dass die Benutzeroberfläche von den gespeicherten Daten getrennt wird. Die Benutzeroberfläche und Sitzung existiert nur auf dem Client und die gespeicherten Daten bzw. zur Verfügung

[1] Vgl. Biehl 2015, S. 15ff.

gestellten Funktionen existieren nur auf dem Server. Somit wird die Portierbarkeit und Skalierbarkeit des Gesamtsystems verbessert. Zudem wird die Möglichkeit einer unabhängigen Weiterentwicklung der verschiedenen Komponenten sichergestellt.[2]

Zustandslosigkeit

Des Weiteren soll eine RESTful-API zustandslos angelegt sein. Das heißt im Genaueren, dass die Kommunikation der verschiedenen Parteien zustandslos sein muss. Es muss für den Server möglich sein, die Anfrage des Clients vollständig zu verstehen bzw. zu verarbeiten, ohne zusätzlich auf vergangene Anfragen zugreifen zu müssen. Für den Client muss es möglich sein, jede Antwort des Servers ohne zusätzliche Informationen zu verstehen, die eventuell zu einem früheren Zeitpunkt angefordert wurden. Das heißt, dass in jeder Anfrage immer alle notwendigen Informationen mitgeschickt werden müssen, von keinem "Vorwissen" ausgegangen werden darf und Sitzungsinformationen ausschließlich auf dem Client gespeichert werden. Durch diese Bedingung verbessert sich die Skalierbarkeit weiter, da der Server keine Ressourcen für die Speicherung der Request-Stati freihalten muss. Zudem steigt die Zuverlässigkeit der Schnittstelle und das Monitoring vereinfacht sich, da bei einem Fehler immer nur ein Request betrachtet werden muss. Somit ist ein Fehler einfacher behebbar und hat keine Auswirkungen auf andere Anfragen.[3]

Caching

Die dritte Designkonvention besagt, dass auf der Client Seite ein Cache vorhanden sein muss. Durch das Markieren von Daten als cache-fähig dürfen die Anfrage-Daten vom Client für spätere identische Requests wiederverwendet werden. Durch dieses Caching von Daten ist es möglich manche Client-Server Interaktionen teilweise oder ganz zu vermeiden, wodurch die Netzwerkauslastung und Skalierbarkeit verbessert wird. Jedoch birgt die Verwendung von Caching das Risiko, dass die Daten im Cache im Vergleich zu den auf dem Server gespeicherten Daten schon veraltet sind, was gegebenenfalls zu Fehlern in der weiteren Verarbeitung führen könnte.[4]

[2]Vgl. Fielding 2000, S. 78.
[3]Vgl. Fielding 2000, S. 78f.
[4]Vgl. Fielding 2000, S. 79ff.

Einheitliche Schnittstelle

Das vierte Prinzip und zentrales Unterscheidungsmerkmal von REST ist das einheitliche Interface zwischen den verschiedenen Komponenten. Hierdurch wird die Systemarchitektur durch das Prinzip der Generalität vereinfacht und die Schnittstelle ist einfacher benutzbar. Zudem wird eine unabhängige Weiterentwicklung der verschiedenen kommunizierenden Komponenten durch die Trennung von Implementierung und Service gewährleistet.
Um ein einheitliches Interface zu erreichen, finden mehrere Beschränkungen auf die Schnittstelle Anwendung: Die Ressourcen der Schnittstelle sollen eindeutig identifizierbar sein. Eine Ressource ist eine vom Interface bereitgestellte Information, die eindeutig über einen URI identifizierbar ist. Die Information hinter der Ressource kann statisch festgelegt oder dynamisch veränderbar sein und muss beim Erstellen der Ressource noch nicht existieren. Das erleichtert die Verarbeitung verschiedener Informationsarten, da auf abstrakter Ressourcenebene nicht zwischen bestimmten Typen unterschieden wird und die benötigte Information auch noch zu einem späteren Zeitpunkt, je nach Inhalt der Anfrage, festgelegt werden kann. Zudem hat jeder Service, der nach den REST Prinzipien entworfen ist eine URL, also eine eindeutige Adresse. Durch diese URL ist der Zugriffsweg zum Webservice standardisiert. Durch diese eindeutig identifizierbaren Ressourcen und Services wird zudem die Kombinierbarkeit verschiedener Ressourcen eines Services bzw. von verschiedenen Services in einem größeren System erleichtert. Eine weitere Beschränkung für die Schnittstelle ist die Verwendung von Repräsentationen zur Veränderung von Ressourcen. Eine Repräsentation ist eine Folge von Bytes, die eine Ressource in einer bestimmten Darstellung, zugehörige Metadaten und Informationen über die Veränderlichkeit abbildet. Somit kann eine Ressource vom Server in verschiedenen Repräsentationen, je nach Anfrage, zurückgegeben werden. Veränderungen der Ressource finden nur über die Repräsentation statt. Des Weiteren sollen Antworten des Servers auf Anfragen selbsterklärend sein. Das heißt, das Standard-Methoden und -Datentypen verwendet werden, um die Ressource zu verändern oder Informationen auszutauschen. Die letzte Beschränkung wird als "Hypermedia as the Engine of Application State" bezeichnet. Hiermit ist gemeint, dass die Interaktion mit einer API dynamisch über Hypermedien abläuft. Somit ist auf der Client-Seite nur Basiswissen über Hypertext nötig und der Client kann vom Server entkoppelt werden, da der Server dem Client neben den angeforderten Informationen dynamisch mögliche Interaktionen zurückgibt.[5]

[5]Vgl. Fielding 2000, S. 81f.

Schichtenarchitektur

Die Systemarchitektur soll zudem in Schichten aufgebaut sein. Das heißt, dass eine Schicht jeweils nur die nächste darunter- und darüberliegende Schicht sieht und mit ihr interagieren kann. Dadurch wird die Komplexität des Gesamtsystems reduziert und die unabhängige Weiterentwicklung der einzelnen Schichten gefördert. Zudem können veraltete Dienste abgekapselt werden. Durch Auslagerung von Funktionalitäten in eigene Schichten verbessert sich die Skalierbarkeit des Systems durch Lastverteilung eines Services auf mehrere Netzwerke oder Prozessoren. Dennoch bringt die Schichtenarchitektur auch Nachteile mit sich. Durch die Kapselung der Dienste und Funktionalitäten in Schichten steigt der Verwaltungs- und Wartungsaufwand des Gesamtsystems. Zudem sinkt auch die Geschwindigkeit, mit der Daten verarbeitet werden, da die Anfrage im Verarbeitungsprozess wesentlich mehr Schnittstellen passieren muss. Dieser Nachteil kann jedoch durch die Verwendung von geteilten Caches in den Zwischenschichten kompensiert werden, da durch diese Caches die Anzahl der zu passierenden Schnittstellen reduziert werden kann. Ein weiterer Vorteil der Schichtenarchitektur ist, dass die Anfragen selektiv von den einzelnen Schichten verändert werden können, da der Inhalt dieser selbst-beschreibend und die Bedeutung der Nachricht für die Zwischenschichten sichtbar ist.[6]

Code on Demand

Die sechste (optionale) Designkonvention von REST besagt, dass Code in Form von Skripten oder Apps über die Schnittstelle vom Client heruntergeladen und ausgeführt werden kann. Dies vereinfacht die Programmlogik des Clients, da weniger Programme schon im Voraus vorhanden sein müssen. Zudem wird dadurch die Erweiterbarkeit eines Systems verbessert, da auch nach dem initialen Installieren eines Systems, dieses noch durch das Bereitstellen von Code über die Schnittstelle erweitert werden kann.[7]

[6] Vgl. Fielding 2000, S. 82f.
[7] Vgl. Fielding 2000, S. 84f.

2.2. ABAP RESTful Application Programming Model

ABAP RAP ist ein Programmiermodell der SAP auf Basis von ABAP, das die Architektur für die Entwicklung von OData-Services definiert. Es basiert auf den Designprinzipien nach REST. Mit diesem Modell können sowohl Web APIs veröffentlicht und Business Events erzeugt als auch Fiori Apps entwickelt werden. RAP kann sowohl in Cloud, als auch in on-premise Systemen eingesetzt werden.[8]

RAP baut im Allgemeinen auf drei Säulen auf: Anders als in vorhergehenden Programmiermodellen, in denen mehrere Tools zum Entwickeln von einer Fiori App nötig waren, sind in RAP alle Implementierungsaufgaben in einer Entwicklungsumgebung integriert, um einen standardisierten Entwicklungsprozess zu gewährleisten. Die zweite Säule ist die Programmiersprache ABAP: Durch Erweiterungen und Anpassungen ist es möglich diese für die Entwicklung mit RAP zu verwenden. Hierbei kommen Technologien wie CDS-Views zum Einsatz, um Datenmodelle zu definieren. Zudem können vorgefertigte APIs für generische Entwicklungsaufgaben verwendet werden. Die dritte Säule sind umfassende Frameworks, die dem Entwickler helfen, effizient und in kurzer Zeit eine Anwendung zu entwickeln, da einzelne Bausteine automatisch generiert werden und an bestimmten Stellen noch anwendungsspezifische Logik eingefügt werden kann.[9]

[8] Vgl. SAP 2023b.
[9] Vgl. SAP 2023b.

Architektur eines Business Services in RAP

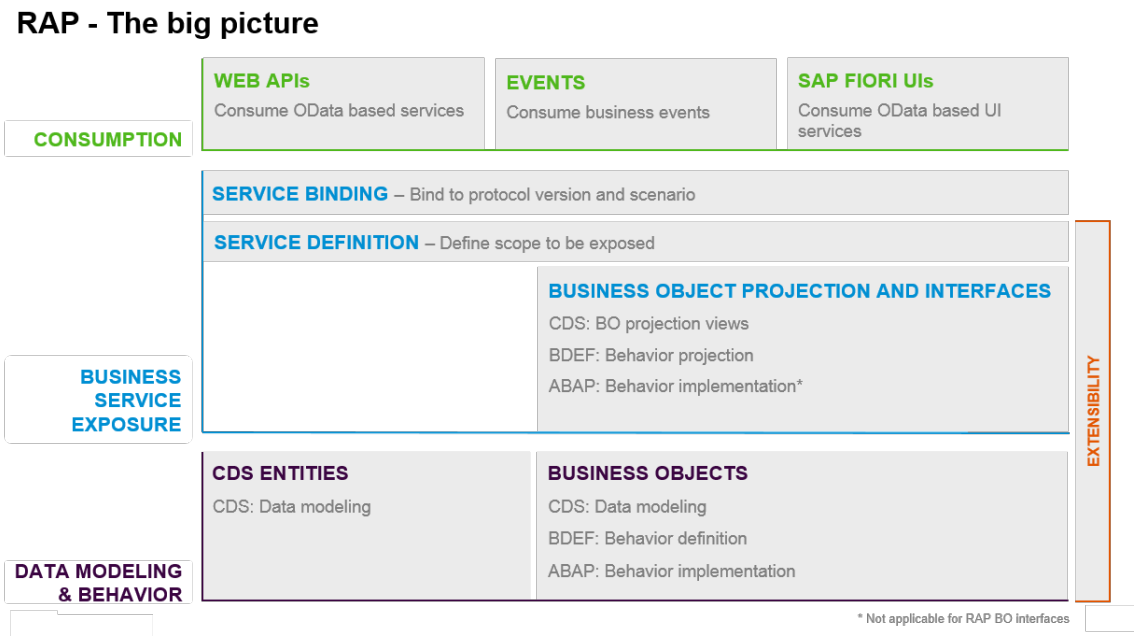


Abbildung 2.: RESTful Application Programming Model Architektur. Abgerufen von SAP 2023b am 09.07.2023.

Modellierung von Daten und Behaviors

Auf der untersten Ebene (Vgl. Abb. 2) werden die benötigen Daten und das beabsichtigte Verhalten modelliert. Dies kann entweder durch CDS-Views oder durch Business Objects geschehen. Core Data Services sind ein Framework um ein Datenmodell, basierend auf der HANA-Datenbank, zu definieren und zu organisieren. Genauer werden alle benötigten Spalten aus einer oder mehreren Tabellen ausgewählt, bei Bedarf gefiltert und diese bei Bedarf mit Annotationen versehen, um speziellere Anforderungen zu erfüllen. Die SQL-Abfrage, um diese Daten von der Datenbank abzurufen ist in dem CDS-View integriert. Der Zweck eines CDS-Views ist jedoch lediglich das Lesen und Strukturieren der Daten; es können hiermit keine Daten verändert werden.[10]

Eine andere Möglichkeit ein Datenmodell zu erzeugen ist durch Business Objects (”BO” abgekürzt). Diese bieten zudem die Implementierung von Behaviors (Verhalten) und einer Laufzeit. Ein BO ist aus struktureller Sicht ein hierarchisch aufgebauter Baum aus mehreren Knoten, die Daten enthalten und durch Eltern-Kind-Beziehungen

[10]Vgl. SAP 2023b.

(sogenannte Kompositionen) miteinander verknüpft sind. Diese Knoten werden durch CDS Entitäten dargestellt. Der hierarchisch oberste Wurzelknoten stellt dabei die Repräsentation des BO an sich dar. Um das Verhalten eines BO zu spezifizieren, muss eine Business Object Behavior Definition ("BD" abgekürzt) angelegt werden. Dieses ABAP Objekt beschreibt das gewünschte Verhalten des BO in RAP. Die BD bezieht sich immer auf die Wurzel-CDS-Entität eines BO und wird als ABAP Klasse implementiert. Ein Behavior beschreibt welche Operationen für ein BO verfügbar sein sollen. Es besteht außerdem noch aus der "Behavior characteristic", die zusätzliche Eigenschaften, wie z. B. Autorisierungen für Operationen festlegt. Operationen sind z. B. create() für das Erstellen, update() für das Aktualisieren und delete() für das Löschen eines Datensatzes. Die Laufzeit eines BO besteht aus zwei Phasen: In der Interaktionsphase werden durch das Ausführen von Operationen Daten gelesen und/ oder verändert. Diese Veränderungen werden zunächst in einem "transactional buffer" (transaktionalen Pufferspeicher) gespeichert und nachdem alle Änderungen durchgeführt wurden in der sog. "save sequence" auf der Datenbank persistiert.[11]

Veröffentlichung des Business Service

Die zweite Ebene (Vgl. Abb. 2) sorgt für das Projizieren von BO und die Erstellung sowie Veröffentlichung von Business Services. Ein Business Service ist in RAP ein RESTful Service, der Repräsentationen von Ressourcen veröffentlicht, die dann von Konsumenten abgerufen werden können. Die Bestandteile eines Business Services werden später noch genauer erläutert. Die Projektion eines BO ist notwendig, um es flexibel konsumieren zu können, da dieses an sich komplett unabhängig vom OData-Service ist. Das BO an sich stellt die maximal möglichen Funktionen und Daten bereit, die Service-unabhängig implementiert und ggf. durch die Projektion auf die für den Service relevanten Aktionen und Daten eingeschränkt werden. Zudem können genauere Anpassungen z. B. im Bezug auf die Darstellung auf einer Benutzeroberfläche über UI-Annotationen erfolgen, die aber nicht Teil des Datenmodells sein sollen. Eine zusätzliche Projektionsschicht hat mehrere Vorteile: Zum einen kann das zugrundeliegende BO angepasst und erweitert werden, ohne dass der darauf aufbauende Service davon betroffen ist. Zum anderen können verschiedene Projektions-Views für verschiedene Anforderungen erstellt werden, die alle dasselbe BO wiederverwenden. Zudem können die Daten und Funktionen eines Services für eine Fiori App oder Web API veröffentlicht werden. Des Weiteren können Services

[11] Vgl. SAP 2023b.

somit auch rollenbasiert veröffentlicht werden, sodass unterschiedliche Daten und Funktionen für unterschiedliche Anwender bereitgestellt werden können. Um eine solche Projektionsschicht zu erstellen, muss zusätzlich ein CDS Projection View erstellt werden, um die speziellen Daten einer Projektion darzustellen. Dieser basiert auf dem CDS View des BO und erzeugt selbst keine neue SQL-View, sondern nur eine Repräsentation der dargestellten Entitäten. Um eine CDS-Entität eines BO zu projizieren, müssen die Wurzel-Entität sowie alle Eltern-Entitäten ebenfalls projiziert sein. Zudem wird auch eine Projection Behavior Definition benötigt, die alle Verhalten, die für einen speziellen Service veröffentlicht werden sollen, projiziert.[12]

Nachdem Teile des BO projiziert wurden und die zugehörigen Artefakte erstellt wurden, muss in der zweiten Ebene ein Service definiert werden. In einer "Business Service Definition" (abgekürzt "BSD") wird festgelegt, welche CDS Entitäten eines Datenmodells, also welche Daten, in einen bestimmten Service veröffentlicht werden sollen. Die BSD stellt eine Protokoll-unabhängige und Konsumenten-spezifische Sichtweise auf das Datenmodell dar. Es können auch mehrere CDS-Entitäten oder eine komplette BO-Struktur in einem Service veröffentlicht werden. Dafür muss in der BSD die hierarchisch höchste Entität des BO, die veröffentlicht werden soll, markiert werden. Diese dient dann als Einstiegspunkt für den Service.[13]

Als letzter Schritt in der zweiten Ebene muss noch das Kommunikationsprotokoll des Service im Service Binding definiert werden. Ein häufiges Beispiel wäre hierbei OData für das Bereitstellen von Daten in einer Fiori App. Ein Service Binding bezieht sich immer direkt auf eine oder mehrere Business Service Definitions. Es können auch mehrere Service Bindings basierend auf einer BSD erstellt werden. Das ist z. B. hilfreich, wenn derselbe Service mit unterschiedlichen Kommunikationsprotokollen veröffentlicht werden soll, da durch die Trennung von Service Definition und Binding das Protokoll von der Geschäftslogik getrennt wird. Somit kann der Entwicklungsaufwand für einen Service erheblich reduziert werden. Ein Service kann grundlegend auf zwei Arten veröffentlicht werden: Entweder als UI Service mit den Protokollen OData oder Information Access, indem man dem durch UI-Annotationen eine Fiori Elements Benutzeroberfläche hinzufügt oder als Web API, die von Clients über das Web konsumiert werden kann. Ein Service kann zudem in mehreren Versionen existieren. Dies geschieht durch das Hinzufügen oder Entfernen von zusätzlichen

[12]Vgl. SAP 2023b.
[13]Vgl. SAP 2023b.

Service Definitions zu einem Service Binding. Damit kann ein Service geändert oder erweitert werden.[14]

Konsumieren des Services

Die dritte und oberste Ebene der RAP Architektur (Vgl. Abb. 2) ist für das Konsumieren der Daten eines Business Services verantwortlich. Es gibt zwei Möglichkeiten, diese Daten durch einen Service zu konsumieren. Die erste Möglichkeit ist durch eine Web API. Die Metadaten eines so veröffentlichten Services enthalten keine Informationen über eine Benutzeroberfläche für die Darstellung der Informationen. Der Zugriff auf die bereitgestellten Daten erfolgt über eine öffentliche Schnittstelle des OData Services. Eine weitere Möglichkeit einen Service zu konsumieren ist innerhalb einer Fiori Elements Anwendung als UI Service. Hier werden die Konfigurationen für die Benutzeroberfläche und das Frontend der Anwendung, die im Backend als Annotationen in den CDS Entitäten festgelegt wurden, über die Metadaten mitgegeben. Somit kann das Fiori Elements Framework aus diesen Metadaten direkt eine fertige UI generieren. Als letzte Möglichkeit ein BO zu konsumieren sind noch Business Events zu nennen. Diese werden hier nur kurz genannt und in einem späteren Kapitel detaillierter beschreiben.[15]

2.3. SAP Fiori Elements

Fiori Elements ist ein Framework zum Entwickeln benutzerfreundlicher und ansprechender Anwendungen. Apps werden auf Basis von OData-Services und UI-Annotationen in den CDS Entitäten durch ein umfassendes Framework fast automatisch generiert. Somit ist im Gegensatz zur älteren SAP UI5 Freestyle Technologie kein JavaScript Coding mehr nötig um das Frontend zu programmieren. Elements benutzt vordefinierte Layouts und Controller für Aktionen der App.[16]

Fiori Elements bietet drei zentrale Vorteile: Es soll dabei helfen, dass sich Entwickler auf die spezifische Geschäftsprozesslogik und die Backend-Entwicklung fokussieren und somit weniger Zeit für die Programmierung der Benutzeroberfläche benötigen, was insgesamt zu einer verkürzten Entwicklungszeit von Apps und somit auch für

[14] Vgl. SAP 2023b.
[15] Vgl. SAP 2023b.
[16] Vgl. SAP 2022a.

niedrigere Entwicklungskosten sorgt. Zudem wird die Kontinuität des UI über alle Fiori Apps hinweg und die Übereinstimmung der Apps mit den SAP Designkonventionen sichergestellt. Die Benutzer der Apps haben so eine einheitliche Benutzungserfahrung (Layout, Navigation, Suche, ...) über alle Apps hinweg. Zudem stellt das zentrale Framework auch sicher, dass der Programmcode für die Benutzeroberflächen der Apps immer sofort funktioniert und bietet außerdem weitere Funktionen wie Übersetzungen und Unterstützung für mobile Endgeräte. Diese Vorteile tragen wiederum zu einer reduzierten Entwicklungszeit und somit einem Kostenersparnis bei.[17]

Vergleich Fiori Elements - SAP UI5 Freestyle

Verglichen mit SAP UI5 Freestyle (Freestyle abgekürzt), der anderen Technologie, um Fiori Apps zu entwickeln, lassen sich folgende Unterschiede feststellen: Die generelle Herangehensweise bei Fiori Elements zielt eher auf das effiziente und schnelle Entwickeln und Veröffentlichen einer App ab. Im Gegensatz dazu, liegt der Fokus bei Freestyle eher auf der Flexibilität, spezielle Anforderungen, die ggf. auch nicht mit dem Elements-Standard übereinstimmen, abbilden zu können. Das wirkt sich auch auf die Möglichkeiten im UI-Design aus: In Fiori Elements ist der Entwickler an die vordefinierten SAP Vorlagen gebunden, während es bei den Freestyle Designs nicht der Fall ist. Das hat aber auch zur Folge, dass bei Freestyle wesentlich mehr Webentwicklungs-Kenntnisse nötig sind, da die Oberfläche mithilfe von JavaScript Coding selbst programmiert werden muss. In Fiori Elements hingegen wird die gesamte Oberfläche vom Elements Framework generiert und muss nur durch UI-Annotationen in den CDS-Entitäten an die Wünsche des Kunden angepasst werden. Was die Wartbarkeit und entwicklungstechnischen Freiheiten der beiden Technologien angeht, sind auch bei Freestyle größere Spielräume vorhanden: Dadurch, dass die Oberfläche selbst programmiert wird, ist es möglich eine Logik in das Frontend einzubauen. Diese Tatsache ist für die Problemstellung der Arbeit sehr relevant, da sequentielle Prozesse, die asynchrone Kommunikation benötigen, sich durch eben diese Logik leicht abbilden lassen. Da in Fiori Elements die UI generiert wird und die Logik von SAP kommt, fehlt diese Gestaltungsmöglichkeit. Dennoch muss man sagen, dass in allen anderen Fällen Fiori Elements erhebliche Vorteile bietet, da bei der Entwicklung einer App sehr viel Zeit und somit auch Geld gespart werden kann.[18]

[17] Vgl. SAP 2022a.
[18] Vgl. SAP 2023c.

3. Praktischer Teil

3.1. Lösungsansätze

3.1.1. Business Workflows

Der erste mögliche Ansatz sind Business Workflows (BW abgekürzt). BWs können benutzt werden, um Geschäftsprozesse im SAP-System abzubilden und decken das Spektrum von einfachen Genehmigungsprozessen bis hin zu komplexen Abläufen ab. Sie eignen sich vor allem für standardisierte Prozesse mit mehreren Bearbeitern. Mit Workflows können durch die Benutzung der bereits bestehenden Funktionen und Transaktionen des SAP-Systems neue Geschäftsprozesse abgebildet werden. In Kombination mit Organisationsmanagement können die einzelnen Schritte des BWs durch bestimmte Akteure ausgeführt werden. Das kann auch auf bestimmte Stellen abstrahiert werden, um von personellen Veränderungen innerhalb des Unternehmens unabhängig zu sein. Workflows können auch untereinander durch das Versenden und Konsumieren von Nachrichten kommunizieren. Diese Kommunikation ist auch zwischen verschiedenen SAP-Systemen über das Internet mit XML-Dokumenten möglich.[1]

Aufbau eines Business Workflows

Zunächst wird die Definition der Aufbau eines Business Workflows beschrieben. Diese lässt sich in vier Bereiche unterteilen.

[1] Vgl. SAP 2022b.

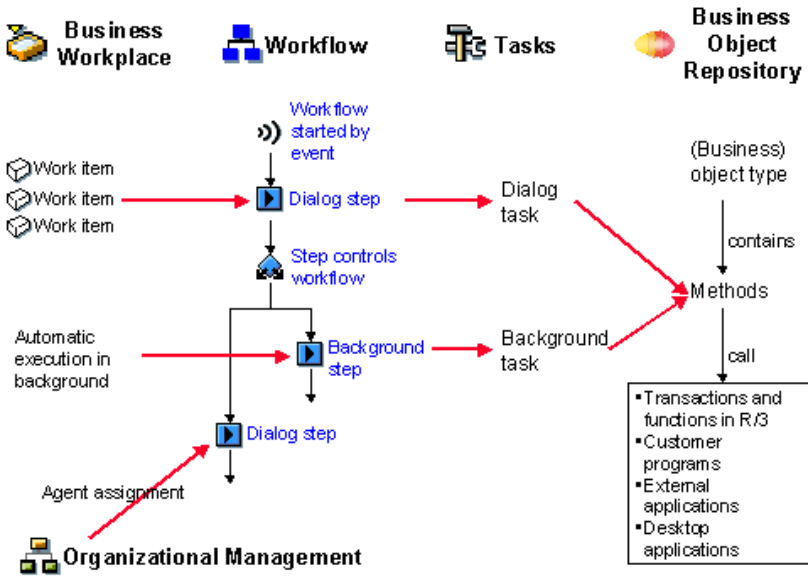


Abbildung 3.: Aufbau eines Business Workflows. Abgerufen von SAP 2022b am 17.07.2023.

Der Business Workplace (Vgl. Abb. 3) ist der Ort in einem SAP-System, in dem der Endanwender "Work Items" (übersetzt aus dem Englischen: "Arbeitspakete" oder "Aufgaben", WE abgekürzt) abhängig vom Zeitpunkt im Geschäftsprozess und den Berechtigungen des Users ausführen kann. Ein WE stellt zur Laufzeit des BWs einen Schritt des Prozesses dar, der ausgeführt wird. Hier werden jedoch keine WEs, die im Hintergrund ausgeführt werden, angezeigt.[2]

Ein Workflow muss vor Ausführung in der "Workflow Definition" (Vgl. Abb. 3) angelegt werden. Diese Definition legt die Reihenfolge der auszuführenden Schritte des Prozesses fest. Zusätzlich können noch Kontrollschritte, Bearbeiter und Fristen für bestimmte Schritte festgelegt werden, die dann zur Laufzeit des Workflows vom "Work Item Manager" verwaltet werden. Es gibt viele Arten von Schritten, die gängigen Konzepten in der Programmierung ähneln, wie z. B. normale Aktivitäten, Fallunterscheidungen, Schleifen. Zudem gibt es Schritte zum Versenden von Nachrichten, Auslösen von Events, Benutzerentscheidungen, usw. Diese Schritte können entweder im Dialog mit einem Benutzer oder automatisch vom System im Hintergrund ausgeführt werden. Ein Workflow kann nicht nur manuell gestartet werden, sondern auch systemseitig von einem bestimmten Event ausgelöst werden.

[2]Vgl. SAP 2022b.

Hierfür muss in der Definition des Workflows das gewünschte Event als Auslöser angegeben werden.

Die einzelnen Schritte, die innerhalb des Workflows ausgeführt werden, heißen Tasks (Vgl. Abb. 3) und stellen grundlegende betriebliche Tätigkeiten dar. Die Dialog- und Hintergrund-Schritte in der Workflow Definition korrespondieren hier mit Dialog- oder Hintergrund-Tasks. Im Workflow bezieht sich ein Task immer auf eine Methode eines Objekttyps, die automatisch ausführbar sein oder aktiv von einem Benutzer gestartet werden können. Eine Methode kann Transaktionen oder Funktionen innerhalb des ERP-Systems aufrufen. Spezielle Anforderungen können durch kundeneigene Logik, oder Schnittstellen zu anderen Systemen umgesetzt werden.[3]

Das "Business Object Repository" (Vgl. Abb. 3) bietet eine Übersicht über alle in einem SAP-System verfügbaren Objekttypen. Man kann die bereits vorhandenen Objekttypen bei Bedarf anpassen oder neue erstellen.[4]

3.1.2. Business Events

Der zweite Ansatz ist die Umsetzung mit Business Events (BE abgekürzt). BEs sind Events die von BOs erzeugt und konsumiert werden können. Dieser eventgesteuerte Kommunikationsansatz, der die asynchrone Kommunikation zwischen dem Event-erzeugenden und -konsumierenden BO ermöglicht, wird von RAP im Standard unterstützt. Hier ist keine direkte Antwort des Empfängers nötig, ein BO erzeugt ein Event und dieses wird von anderen BOs dann weiterverarbeitet, ohne dass der Erzeuger weiterhin in diesen Prozess involviert ist. Ein BE stellt eine signifikante Veränderung eines BOs dar, die im Zuge eines Behaviours erzeugt wird. Dem Event werden dann über dessen Metadaten alle nötigen Informationen, anhand derer die Weiterverarbeitung, je nach speziellem Anwendungsfall, stattfindet, mitgegeben.[5]

Ein BO kann als Event-Erzeuger oder Event-Konsument auftreten. Zuerst wird die Erzeuger-Seite betrachtet.

[3]Vgl. SAP 2022b.
[4]Vgl. SAP 2022b.
[5]Vgl. SAP 2023d.

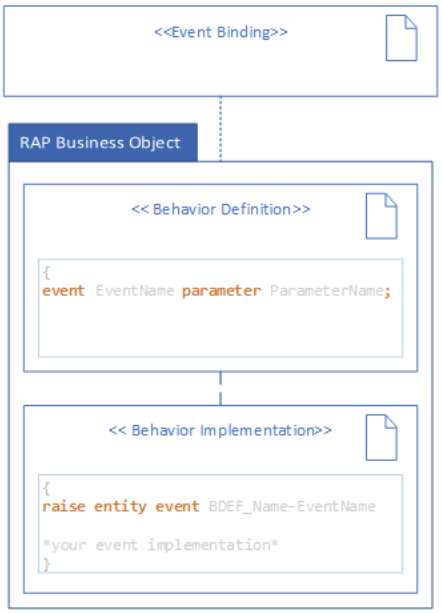


Abbildung 4.: Business Object als Event-Erzeuger. Abgerufen von SAP 2023d am 17.07.2023.

Ein Event wird in der Behaviour Definition (Vgl. Abb. 4) eines BO erstmals definiert. Hier können dann Parameter für die weitere Verarbeitung in den Metadaten des Events mitgegeben werden. Danach kann es in der dazugehörigen Behavior Implementation innerhalb einer speziellen Operation des BO implementiert werden. Ein Event wird in der save-sequence der RAP Laufzeit erzeugt, nachdem die Änderungen auf der Datenbank persistiert wurden. Durch das Event Binding wird das Event noch einem speziellen Namensraum, BO und zugehöriger Operation zugeordnet.[6]

BOs können BEs nicht nur erzeugen, sondern auch verarbeiten. Dies geschieht über das Event Consumption Model. Ein Event Consumption Model besteht aus einer Reihe von ABAP-Artefakten, mit denen man Events im SAP-System konsumieren kann. Diese Events müssen jedoch dem Standard eines Cloud Events entsprechen, da der Austausch über das SAP Event Mesh abgewickelt wird, was ein Dienst der SAP Business Technology Platform und somit komplett cloud-basiert ist. Hierfür muss ein inbound bzw. outbound Event-Binding für ein- bzw. ausgehende Events erstellt werden. Diese Bindings sind Teil des SAP Enterprise Event Enablement Frameworks, das den Austausch von Events zwischen dem S/4 System und der BTP regelt.[7]

[6] Vgl. SAP 2023d.
[7] Vgl. SAP 2022c.

3.1.3. Background Processing Framework

Die letzte Möglichkeit, die betrachtet wird, um asynchrone Prozesse abzubilden, stellt das ”background Processing Framework” (bgPF abgekürzt) dar.

Das bgPF ist ein Framework, das die Möglichkeit schafft asynchron zum Hauptprozess Logik auszulagern und auszuführen. Zuerst soll auf die grundlegende Funktionsweise des bgPF eingegangen werden.

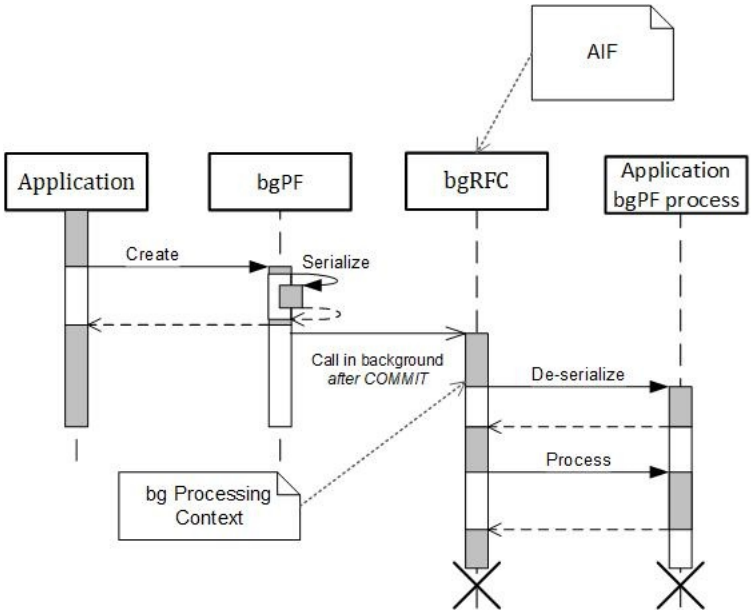


Abbildung 5.: Funktionsweise des bgPF. Abgerufen von Anhang A.3 am 17.07.2023.

Nachdem das Interface für das bgPF in der Anwendung als eigene Klasse mit der jeweiligen Logik, die ausgelagert werden soll, implementiert wurde, kann es verwendet werden. Die Anwendung ruft dann zur Laufzeit diese Klasse auf und erstellt synchron ein bgPF-Objekt. Innerhalb dieses Objekts wird die Klasse dann serialisiert und für die spätere Ausführung gespeichert. In der Endphase des Hauptprozesses wird dann mit einem bgRFC, also einem Funktionsaufruf, der asynchron im Hintergrund ausgeführt wird, die Ausführung ausgelöst. Der bgRFC wird dann in einer neuen ABAP Laufzeit als neuer Prozess gestartet. Hier wird die serialisierte Klasse wieder deserialisiert und die implementierte Logik ausgeführt (Vgl. Abb. 5). Falls nötig, können durch das Bereitstellen eines eigenen Verarbeitungskontexts von der aufrufenden Anwendung spezifische Verarbeitungsprioritäten festgelegt werden. Dies ist zwingend notwendig, wenn eine Verarbeitungsreihenfolge eingehalten werden muss. Der voreingestellte

Kontext wird standardäßig von allen Anwendungen gemeinsam verwendet. (vgl. Anhang A.3, Seite 57ff)

Das Entkoppeln und asynchrone Ausführen gewisser Logik vom Hauptprozess bringt einige Vorteile mit sich: Zuerst wird die Robustheit und transaktionale Konsistenz des Hauptprozesses und somit auch die Konsistenz der Daten sichergestellt. Das bgPF bietet die Möglichkeit, diese Konsistenz zu gewährleisten, aber auch gleichzeitig relevantes Coding aus dem Hauptprozess auszulagern, was dessen Robustheit verbessert. Des Weiteren erscheint die Hauptanwendung für den Benutzer performanter, da gewisse Verarbeitungslogik in einer anderen Sitzung asynchron ausgeführt werden kann. Ein weiterer Vorteil ist die bessere Skalierbarkeit, wenn größere Prozesse in kleinere Teile unterteilt werden, die verteilt über mehrere Laufzeitumgebungen asynchron ausgeführt werden können. Die Skalierung einer Anwendung kann mit dieser Technologie dann komplett vom SAP-System automatisiert werden. Diese Funktionalität des automatischen Skalierens befindet sich jedoch zum Zeitpunkt des Verfassens dieser Arbeit noch in der Entwicklung und ist noch nicht produktiv verfügbar. (vgl. Anhang A.3, Seite 57ff)

bgPF innerhalb von RAP

Das bgPF kann auch innerhalb von RAP verwendet werden. Konkret wird die Methode des bgPF-Objekts, die das Speichern für die spätere asynchrone Ausführung übernimmt, in der zweiten Phase der Laufzeit des BOs aufgerufen und das Objekt somit auf der Datenbank gespeichert, von wo aus die Logik in einer anderen Sitzung mittels eines bgRFC ausgeführt wird. (vgl. Anhang A.3, Seite 57ff)

3.2. Vergleich der Ansätze

Im Folgenden sollen die in den vorherigen Kapiteln vorgestellten Ansätze miteinander verglichen werden. Hierfür werden die Kriterien Komplexität der Implementierung, also welchen Aufwand das Umsetzen der Technologie im Anwendungsfall verursacht, die Auswirkungen auf die Systemlandschaft, also ob zusätzliche Systemkomponenten benötigt werden und die Performance der einzelnen Ansätze im Bezug auf Ausführungsdauer und Rechenlast verglichen. Weitere Kriterien sind zusätzliche Kosten die durch eine Lösung eventuell entstehen, die Flexibilität der Technologien im Bezug auf Gestaltungsmöglichkeiten von Anwendungsszenarien und Integration mit

anderen Technologien/ Frameworks und die Skalierbarkeit. Abgeschlossen wird der Vergleich mit einer Betrachtung der Wartbarkeit der drei Umsetzungsmöglichkeiten und Abwärtskompatibilität zu älteren Systemen.

Komplexität der Implementierung

Als erstes Kriterium wird die Komplexität der Implementierung der drei Technologien herangezogen. Genauer wird hier auf die Anzahl der zu erstellenden Artefakte, eine Beschreibung der Implementierungstätigkeiten und - wenn möglich - eine ungefähre Lines-of-Code Angabe betrachtet.

Die Komplexität von Workflows hängt von dem Geschäftsprozess ab, der abgebildet werden soll. Somit ist das Spektrum von einem sehr einfachen Workflow, der beispielsweise bei einem gewissen Ereignis einen Mitarbeiter per E-Mail benachrichtigt bis hin zu einem Workflow für einen komplexen Geschäftsprozess, wie einer Abwesenheitsmeldung, bei dem viele Schritte mit Genehmigungsprozessen nötig sind und der in viele Stellen im System hineingreift, möglich. Durch die freien Gestaltungsmöglichkeiten eines Workflows, einen Prozess aus beliebig vielen Schritten in beliebiger Konstellation zu modellieren sind Workflows in Produktivsystemen meistens sehr komplex. Hinzu kommt die weitgehenden Konfigurationsmöglichkeiten eines jeden Schrittes in einem Workflow und die Möglichkeit eigenes ABAP-Coding in einem Schritt auszuführen. Hierdurch kann die Komplexität nochmals stark steigen. Zudem können Workflows, wie oben angesprochen ereignisgesteuert gestartet werden und miteinander interagieren, wodurch komplexe Wechselwirkungen zusätzlich beachtet werden müssen. Somit lässt sich insgesamt sagen, dass Workflows zwar potenziell simpel sein können, jedoch in der Realität eher sehr komplex sind. (vgl. Anhang A.2.1, Z. 22-32)

Bei Business Events hängt die Komplexität vom dem verwendeten Ansatz ab. Beispielsweise sind Standard Events von SAP sehr einfach, diese sind bereits im System vorhanden und müssen nur über eine Einstellung eingeschaltet werden. Normalerweise werden die Events bereits von der SAP entwickelt und an die Kunden ausgeliefert. Wenn jedoch für eine spezielle Anforderung ein eigenes Event benötigt wird, kann auch ein Custom Event erstellt werden. Diese werden entweder RAP-basiert entwickelt, was einen nicht unerheblichen Aufwand darstellt und Erfahrung in der Entwicklung mit RAP voraussetzt. Die andere Option ist das Netweaver Event Enablement Add-On, womit man durch einen Low-Code-Ansatz ein Event einfach in

kurzer Zeit konfigurieren kann. Insgesamt ist festzustellen, dass der Aufwand von wenigen Sekunden mit dem Aktivieren eines Standard Events bis hin zu mehreren Tagen mit einem selbst programmierten RAP-basiertem Event reicht und unterschiedliche Artefakte erstellt werden müssen. (vgl. Anhang A.2.2, Z. 28-63)

Die Komplexität bei der Implementierung des bgPF beschränkt sich auf die Implementierung einer der zur Verfügung stehenden Interfaces. Ab hier übernimmt dann das Framework das Serialisieren und Weitergeben der Klasse an das bgRFC Framework, das dann für die asynchrone Ausführung zuständig ist. Die Komplexität des bgRFC Frameworks beschränkt sich für den Anwendungsprogrammierer, nachdem es initial beim Einrichten des Systems konfiguriert wurde, auf wenige Zeilen Code, in denen die Einheit, die den Funktionsaufruf ausführt, aufgerufen wird. Hier ist die Komplexität jedoch wieder vom konkreten Anwendungsfall abhängig, da die Logik, die im Hintergrund ausgeführt werden soll, vom Entwickler selbst entwickelt werden muss. Die Komplexität hängt also von der Geschäftsprozesslogik ab, die umgesetzt werden soll. Die Konfiguration des bgRFC Framework besteht im Groben aus dem Festlegen auf welchen Applikationsservern eine bgRFC-Einheit ausgeführt werden soll und wie viele Ressourcen diese verbrauchen darf. Insgesamt kann man sagen, dass die Komplexität, abgesehen von der konkreten Anwendungslogik sich auf wenige Konfigurationen und Zeilen Code beschränkt und somit als eher leicht eingestuft werden kann. (vgl. Anhang A.2.3, Z. 17-30)

Auswirkungen auf die Systemlandschaft

Die Auswirkungen auf die Systemlandschaft werden daran bemessen, ob für die Implementierung der jeweiligen Technologie zusätzliche Systemkomponenten, im Spezielleren das Einbinden von Cloud-Komponenten, oder das Restrukturieren von Prozessen notwendig sind.

Workflows sind in der sogenannten ”SAP-Basis” enthalten. Die Systemkomponente ”SAP-Basis” bildet die technische Grundlage eines SAP-Systems, indem sie Systemadministration, Datenbankmanagement, Kommunikation, Sicherheit und Performance-Optimierung gewährleistet und somit eine stabile und effiziente Plattform für die Ausführung der SAP-Anwendungen bereitstellt. Das heißt, dass sie in jedem SAP-System als Grundfunktionalität verfügbar sind. Es müssen keine zusätzlichen Systemkomponenten integriert werden und Workflow sind sowohl in on-premise als auch in

Cloud-Systemen verfügbar. Somit lässt sich zusammenfassen, dass Workflows keine Auswirkungen auf die Systemlandschaft haben. (vgl. Anhang A.2.1, Z. 33-40)

Das Verwenden von Business Events hat auf die Systemlandschaft definitiv mehrere Auswirkungen. Zum einen müssen die Prozesse des Unternehmens selbst angepasst werden, da man von synchronen batchorientierten Prozessen zu asynchronen Prozessen umsteigt. Zudem wird ein Event Broker oder Event Mesh (Netzwerk mehrerer Event Broker) als zusätzliche Systemkomponente benötigt, das den Austausch der Events regelt. Das SAP Event Mesh ist zudem eine Cloud-Komponente, was eine relevante Änderungen für reine on-premise Systemlandschaften darstellt. Es ist aber auch möglich einen Event Broker lokal zu betreiben, wenngleich dann die Vorteile der globalen Vernetzung mit Events verloren gehen. Außerdem lohnt sich diese Option erst wenn eine eventgesteuerte Architektur in wirklich großem Umfang umgesetzt wird. Zudem können mit SAP-Technologie auch Events anderer Hersteller, die dem Cloud Event Standard entsprechen verarbeitet werden, was für heterogene Systemlandschaften mit Systemen Lösungen mehrerer Hersteller ein großer Vorteil ist. (vgl. Anhang A.2.2, Z. 64-108)

Auch das bgRFC Framework ist in der Systemkomponente SAP-Basis enthalten und ist somit bei jedem SAP System vorhanden und einsetzbar. Auch das Ausführen der remote function calls im Hintergrund passiert komplett lokal im System. Somit hat das bgRFC Framework an sich, wie Workflows keine Auswirkungen auf die Systemlandschaft. Das bgPF ist aber als darauf aufbauende Technologie zum heutigen Stand nur für Cloud Systeme verfügbar. Dies stellt aber eher einen Vorteil für Cloud Systeme als einen Nachteil für on-premise Systeme dar, weil die bgRFC Technologie nicht direkt in der Cloud verfügbar ist, sondern nur über den Umweg des bgPF. Zudem wird diese auch mit der Version von 2023 für on-premise Systeme verfügbar werden. Somit ergeben sich beim bgPF grundsätzlich keine Auswirkungen auf die Systemlandschaft, es muss lediglich zum jetztigen Zeitpunt bei Cloud Systemen das bgPF und bei on-premise Systemen der bgRFC benutzt werden.(vgl. Anhang A.2.3, Z. 31-49)

Performance

Die Performance der drei Lösungsansätze wird anhand der Rechenlast, die auf dem System durch das Ausführen des Prozesses erzeugt wird, der Ausführungsdauer und der Netzwerklast die dadurch entsteht gemessen.

Allgemein lässt sich sagen, dass der Performance-Aspekt bei Workflows in einer typischen Systemlandschaft kein Problem darstellt und auch bei sehr häufig stattfindenden Prozessen keine Leistungsgrenzen erreicht werden. Zudem werden Workflows, bedingt durch die Prozesse, die mit ihnen abgebildet werden, meist nicht auf einmal ausgeführt. Meist werden nur einzelne oder wenige Schritte, die für sich keine nennenswerte Last auf dem System erzeugen, direkt hintereinander ausgeführt. Der limitierende Faktor hingegen ist eher der Anwender, da in vielen Prozessen Benutzer manuell Entscheidungen treffen müssen, wodurch die Ausführung verzögert wird. Auch in dem Fall, dass viele Schritte eines Workflows hintereinander ausgeführt werden, bearbeitet das System alle Tätigkeiten im Hintergrund ohne den Anwender zu beeinflussen. Kommunikation über Systemgrenzen hinaus, die die Performance verschlechtern würde, ist zwar theoretisch möglich, aber in der Realität meist nicht notwendig und die Ausführung von Workflows findet beim Großteil der Geschäftsprozesse rein lokal statt. Somit sind Workflows insgesamt eine performante Technologie. (vgl. Anhang A.2.1, Z. 41-66)

Der Einsatz von Business Events ist für das Quellsystem der Events von erheblichem Vorteil, da durch den eventgesteuerten Ansatz die nachgelagerte Code-Ausführung gezielt ausgelöst werden kann und keine Last durch permanente API-Calls, um zu prüfen, ob das relevante Ereignis eingetreten ist, entsteht. Das ausschlaggebende Kriterium bei Business Events im Bezug auf die Performance sind die mitgeschickten Daten: Je mehr Daten bei einem Event mitgeschickt werden, desto größer wird es und desto mehr Last entsteht bei der Erzeugung und Weiterleitung. Hier gilt es, den Mittelweg zwischen der Menge an mitgeschickten Informationen und der Menge an API-Calls zu finden. Insgesamt dürfe das für den konkreten Anwendungsfall jedoch kein Problem darstellen, da kleine Benachrichtigungs-Events ausreichen, um eine asynchrone Code-Ausführung im Hintergrund anzustoßen. (vgl. Anhang A.2.2, Z. 109-135)

Da mit dem bgPF bzw. bgRFC asynchrone Technologien betrachtet werden, entsteht bei einem reinen Zeitaspekt aufgrund der Natur der Frameworks immer eine gewisse Verzögerung bei der Ausführung der speziellen Logik. Wie groß diese Verzögerung dann letztendlich ist, hängt wesentlich von den Konfigurationen des bgRFC ab, das auch beim bgPF für die Ausführung des Codes zuständig ist. Hier ist vor allem wichtig wie viele Ressourcen dem Framework zugewiesen werden, denn je mehr Rechenleistung zur Verfügung steht, desto schneller können die Funktionsaufrufe ausgeführt werden. Zudem kann konfiguriert werden, wie viele Applikationsserver für die Ausführung

solcher Hintergrundtätigkeiten genutzt werden. Dies gilt jedoch nur für den bgRFC und nicht für das bgPF-Framework. Hier sollten jedoch Wechselwirkungen mit anderen Systemkomponenten beachtet werden, da das restliche System beeinträchtigt werden kann, wenn für andere Prozesse nicht mehr genug Leistung zur Verfügung steht. Somit lässt sich zusammenfassen, dass aufgrund der Ausführung im Hintergrund zwar immer eine gewisse Verzögerung in Kauf genommen werden muss, diese aber durch die Zuteilung von mehr oder weniger Ressourcen steuerbar ist. (vgl. Anhang A.2.3, Z. 50-61)

Kosten

Zunächst wird die monetäre Seite der drei Technologien betrachtet. Es soll die Frage beantwortet werden, ob durch die drei Technologien zusätzliche Lizenzkosten oder laufende Kosten durch Abonnements entstehen.

Da die Workflows in der SAP-Basis mit jedem System enthalten sind, entstehen für den Kunden keine weiteren Kosten durch die Implementierung dieser Technologie. (vgl. Anhang A.2.1, Z. 67-72)

Die Implementierung von Business Events verursacht Kosten. Zum einen durch das Betreiben eines Event Brokers, wie z. B. dem SAP Event Mesh. Kleinere Broker wie dieser, die vor Allem auf kleinere Notification-Events ausgelegt sind, werden gemietet und nach dem Benutzungsvolumen bezahlt. Hier liegen die Kosten, wenn nur einzlene Anwendungsfälle oder Bereiche abgebildet werden im dreistelligen Bereich pro Monat. Soll jedoch die gesamte Unternehmensarichtektur mit Events gesteuert werden, müssen umfangreichere Technologien zum Einsatz kommen, bei denen die Kosten schnell im sechsstelligen Bereich liegen. Hier ist der relevante Faktor die Betrachtung des Break-even-Punkts, ab dem man Geld spart, da die Infrastruktur vorhanden ist und Skaleneffekte eintreten. Ab welcher Anzahl an Events diese Technologie günstiger ist, muss im Einzelfall betrachtet werden. Im konkreten Anwendungsfall sollte das nicht gegen Business Events als Lösungsansatz sprechen, da hier nur ein spezieller Anwendungsfall mit kleinen Events betrachtet wird, der auch mit dem SAP Event Mesh abgebildet werden kann. (vgl. Anhang A.2.2, Z. 136-163)

Bei dem bgPF fällt die Bewertung des Kostenfaktors, ähnlich wie bei Workflows, sehr kurz aus, da auch hier die relevante Technologie bereits in der SAP Basiskomponente enthalten ist und somit in jedem SAP System benutzbar ist. Hier muss zwar der Vollständigkeit halber angemerkt werden, dass man sich in Cloud Systemen für das

bgPF und in on-premise Systemen für das bgRFC Framework entscheiden muss, was aber keine Einschränkungen bedeutet, da beide dieselbe Funktionalität bereitstellen. Auch wenn das Cloud System laufende Kosten verursacht, so entstehen diese aufgrund des Gesamtsystems und nicht aufgrund der Verwendung des bgPF. Somit entstehen hier keine zusätzlichen Kosten für den Anwender. (vgl. Anhang A.2.3, Z. 62-65)

Flexibilität

Beim Kriterium der Flexibilität soll die spätere Anpassbarkeit der Lösungsansätze, die Gestaltungsmöglichkeiten, die sie bei spezielleren Anforderungen bieten und die Integrationsmöglichkeiten zu anderen Technologien betrachtet werden.

Bei der Flexibilität von Workflows muss zwischen klassischen und flexiblen Workflows unterschieden werden. Bei der Betrachtung der Gestaltungsmöglichkeiten bei speziellen Anforderungen verhalten sich beide Varianten gleich, die Vorlagen, die hier von SAP an Kunden ausgeliefert werden, sind nochmals an die konkreten Bedürfnisse und Geschäftsprozesse anpassbar. Im Bezug auf die spätere Anpassbarkeit sind flexible Workflows im Vorteil, da hier Fehlerkorrekturen im Gegensatz zu klassischen Workflows automatisch im Kundensystem übernommen werden, sobald diese von der SAP veröffentlicht wurden. Bei klassischen Workflows müssen diese noch aktiv vom Kunden umgesetzt werden. Jedoch sind flexible Workflows nur für Cloud-Produkte verfügbar. Die Integrationsmöglichkeiten mit anderen Technologien sind für Workflows auch sehr umfangreich, da auf diese als Teil der SAP-Basis von allen anderen Softwarekomponenten aus zugegriffen werden kann. Insgesamt sind Workflows sehr flexibel, da der Kunde nicht an die Vorlagen der SAP gebunden ist und durch das Ausführen von eigenem ABAP-Coding weitgehende Freiheiten hat. (vgl. Anhang A.2.1, Z. 73-115)

Die Flexibilität von Business Events hängt stark von der verwendeten Technologie ab. Auf der einen Seite gibt es Standard-Events, die sehr unflexibel sind, auf der anderen Seite sind Custom Events mit dem Netweaver Event-Enablement Add-On sehr flexibel und schnell anpassbar. Durch die Möglichkeit ein eigenes Event mit RAP zu programmieren erhält der Entwickler fast unbegrenzte Freiheiten, solange man sich im Cloud Events Standard bewegt. Events lassen sich auch untereinander mit anderen Events kombinieren. Diese können z. B. beim Passieren des Event Brokers verändert werden oder in anderen Systemkomponenten gewisse Logik auslösen. Somit sind Business Events insgesamt im Bezug auf ihre spätere Anpassbarkeit und ihre

Integrationsmöglichkeiten mit anderen Technologien sehr flexibel. (vgl. Anhang A.2.2, Z. 164-186)

Die Aufgabe des bgPF-/ bgRFC-Frameworks ist es lediglich, beliebigen Code asynchron im Hintergrund auszuführen. Die Anpassbarkeit dieses Codes und die Möglichkeiten bei speziellen Anforderungen sind somit auf der rein funktionalen Ebene unabhängig vom ausführenden Framework an sich. Somit kann hier resümiert werden, dass die Umsetzbarkeit spezieller Anforderungen und Anpassbarkeit lediglich durch die Möglichkeiten, die ABAP als Programmiersprache zur Verfügung stellt, begrenzt wird und somit kein Problem darstellen sollte. Die Integrationsmöglichkeiten mit anderen Technologien sind genauso wie bei Workflows sehr umfangreich, da auch auf den bgRFC als Teil der SAP-Basis von allen anderen Komponenten aus zugegriffen werden kann und das bgPF in der Cloud auch durch das Implementieren eines Interfaces überall einsetzbar ist. (vgl. Anhang A.2.3, Z. 66-79)

Skalierbarkeit

Die Skalierbarkeit der Komponenten beschreibt ihre Fähigkeit bei wachendem Nutzungsvolumen und Datenlast effizient zu wachsen und die Leistung beizubehalten.

Workflows sind an sich große zusammenhängende Prozesse, die aufgrund der sequentiellen Reihenfolge der einzelnen Schritte nicht unterteilt und z. B. parallel ausgeführt werden können. Somit skalieren Workflows nur linear, die Menge an beanspruchten Ressourcen steigt proportional zur Anzahl der laufenden Prozesse. (vgl. Anhang A.2.1, Z. 116-127)

Das Thema Skalierbarkeit ist eine der großen Stärken von Business Events. Event Broker sind darauf ausgelegt bis zu mehreren Milliarden von Events pro Tag zu verarbeiten. Bei dieser Technologie erreichen als Erstes das Back-End-System, von dem aus die Events losgeschickt werden oder das System, das die Events letztendlich empfängt, ein Limit. Aber auch dieses Limit wird erst sehr spät erreicht, da die Technologie allgemein auf starke Skalierung ausgelegt ist. (vgl. Anhang A.2.2, Z. 187-193)

Beim Kriterium der Skalierbarkeit hat das bgPF noch einige Defizite gegenüber dem bgRFC. Hier ist die Ausführung in lediglich einem Applikationsserver möglich, was die Ressourcenzuteilung je nach Aufgaben und den Last-Ausgleich zwischen den verschiedenen Applikationsservern erschwert. Wenn ein on-premise System

vorhanden ist, ist es möglich verschiedene Applikationsserver für die Ausführung von Logik im Hintergrund zu benutzen und diesen auch unterschiedlich viele Ressourcen einzuräumen. Somit ist hier der bgRFC im Vorteil gegenüber dem bgPF, was zum jetzigen Entwicklungszeitpunkt noch diese Defizite im Bezug auf die Skalierbarkeit aufweist. (vgl. Anhang A.2.3, Z. 80-92)

Wartbarkeit

Software wird als leicht wartbar bezeichnet, wenn sie leicht verändert werden kann, um Fehler zu beheben oder neue Anforderungen zu erfüllen. Dieses Kriterium wird anhand der Analysemöglichkeiten, die für eine Technologie zur Verfügung stehen und ob notwendige Anpassungen an einer Stelle oder über mehrere Systeme verteilt erfolgen müssen, bewertet.

Workflows werden im SAP-System zentral an einer Stelle erstellt, konfiguriert und verwaltet. Von dieser Stelle kann auch zentral das Debugging stattfinden, auch wenn andere Systemkomponenten beeinflusst werden. Somit kann die Wartung als relativ einfach beschrieben werden. (vgl. Anhang A.2.1, Z. 128-133)

Die Wartbarkeit von Business Events wird maßgeblich davon beeinflusst, inwieweit die Grundsätze der eventgesteuerten Architektur bei der Implementierung eingehalten wurden. Dann kann die gesamte Wartung nur an der Eventquelle geschehen, da die Events unabhängig von ihrem Konsumenten konstruiert sind. Realistisch ist der Wartungsaufwand und die Komplexität jedoch etwas höher, da oft die Konsumenten schon im Voraus aus Effizienzgründen Annahmen über die Gestalt der Events treffen. Somit muss die Wartung dann neben dem Quell-System auch beim Konsumenten und dem Event Broker erfolgen. Auch die Art der Events ist entscheidend, da kleinere Notification Events grundsätzlich einfacher zu warten sind als große Daten Events. Insgesamt sind Business Events jedoch gut wartbar. (vgl. Anhang A.2.2, Z. 194-212)

Da das bgPF durch das Implementieren eines Interfaces benutzt werden kann, ist hier die gesamte Wartbarkeit für die Anwendungsfälle direkt im Code oder über den ABAP Debugger möglich und somit sehr einfach. Für die darunterliegende Technologie gibt es einen Transaktionscode von dem aus das bgRFC Framework komplett konfiguriert werden kann. Somit kann die Wartbarkeit als relativ einfach eingeordnet werden. (vgl. Anhang A.2.3, Z. 93-100)

Abwärtskompatibilität

Zuletzt wird noch die Abwärtskompatibilität der Ansätze betrachtet.

Klassische Workflows sind keine neue Technologie mehr und funktionieren in allen gängigen SAP-Systemen. Auf flexible Workflows trifft das jedoch nicht zu, da diese für die Cloud entwickelt wurden und eine neuere Technologie sind. Bei klassischen Workflows wurde in neueren Versionen lediglich die Benutzeroberfläche angepasst, die Funktionalität ist gleich geblieben. Somit können Workflows, die in neueren Versionen entwickelt wurden, problemlos in ältere Systeme kopiert werden. Zusammenfassend eigenen sich Workflows sehr gut, wenn Abwärtskompatibilität ein wichtiges Kriterium ist. (vgl. Anhang A.2.1, Z. 134-143)

Die Abwärtskompatibilität wird auch im Wesentlichen von der verwendeten Technologie bestimmt. Sollen Standard Events zum Einsatz kommen, muss das S/4 Back-End-System immer auf der aktuellsten Version sein, da ältere Versionen die neuen Features nicht unterstützen. Genau für ältere Systeme gibt es das Event Enablement Add-On, was dafür ausgelegt ist, auch noch mit alten SAP ECC Systemen zu arbeiten. Somit sind Business Events zumindest teilweise abwärtskompatibel.(vgl. Anhang A.2.2, Z. 213-223)

Das bgPF ist noch eine sehr neue Technologie, die auch im Moment noch weiterentwickelt wird. Somit ist hier eine Abwärtskompatibilität nicht wirklich gegeben. Dieser Faktor an sich ist jedoch unerheblich, da das Framework sowieso nur in der Cloud verfügbar ist und die Systeme der Kunden hier ohne ihr Zutun immer die neueste Version haben. In einer on-premise Landschaft stellt die Abwärtskompatibilität kein Problem dar, da die bgRFC Technologie schon seit 2005 existiert und somit mit dem Großteil der produktiven Systeme kompatible sein sollte. Insgesamt ist somit im on-premise Umfeld eine sehr gute Abwärtskompatibilität gegeben, in der Cloud jedoch nicht. (vgl. Anhang A.2.3, Z. 101-112)

3.3. Entscheidungsmatrix

Hier soll eine Entscheidungsmatrix entwickelt werden, um die Entscheidungsfindung zu vereinfachen. Die verschiedenen Kriterien, werden im in einem Diagramm pro Technologie für jedes Kriterium auf einer relativen Zahlenskala von 1 bis 5 bewertet, um ihre Stärken und Schwächen besser herauszustellen.

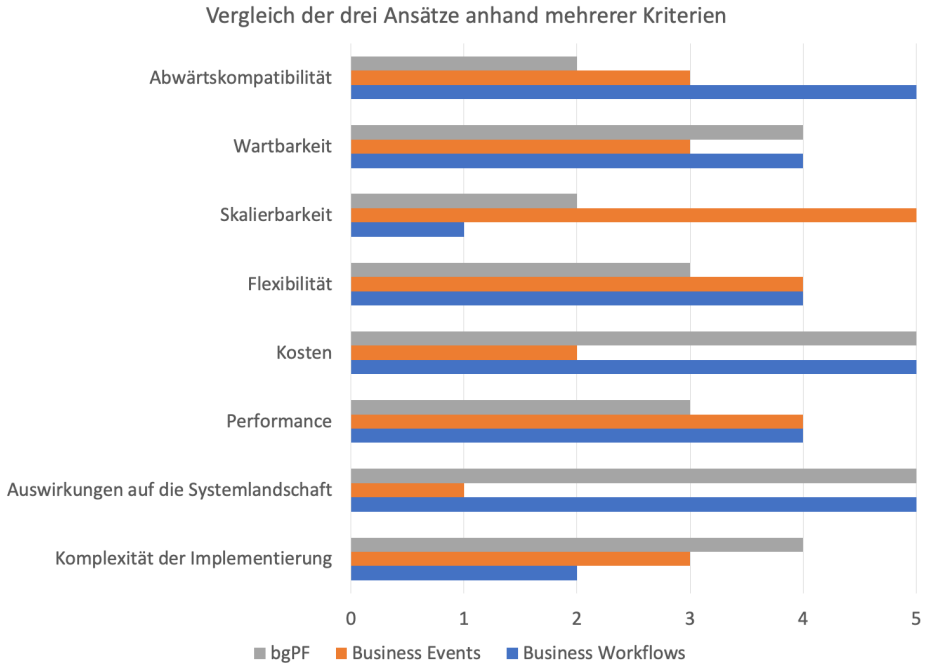


Abbildung 6.: Vergleich der drei Ansätze anhand mehrerer Kriterien. Eigene Darstellung

Das Diagramm veranschaulicht die Stärken und Schwächen der einzelnen Lösungsansätze. Hierbei wird keine Gewichtung der einzelnen Kriterien vorgenommen, sondern die Bewertung der drei Technologien bei jedem Kriterium für sich dargestellt. Workflows schneiden besonders gut bei den Kriterien Abwärtskompatibilität, Kosten und Auswirkungen auf die Systemlandschaft ab, sind dafür aber schwächer in den Punkten Skalierbarkeit und Komplexität der Implementierung (Vgl. Abb. 6). Business Events sind sehr gut skalierbar, gut in den Punkten der Performance und Flexibilität, dafür haben sie aber erhebliche Auswirkungen auf die Systemlandschaft. Das bgPF Framework hat seine Stärken in den Punkten Auswirkungen auf die Systemlandschaft und Kosten, dafür aber Schwächen in Kategorien Abwärtskompatibilität und Skalierbarkeit. Insgesamt sind jedoch alle drei Technologien im konkreten Anwendungsszenario nutzbar.

4. Schlussbetrachtungen

4.1. Zusammenfassung

Die wissenschaftliche Arbeit beschäftigt sich mit der Umsetzung sequentieller Prozesse im RESTful-Umfeld und vergleicht verschiedene praktische Lösungsansätze in SAP-Systemen.

Der theoretische Teil der Arbeit führt in die Grundlagen von RESTful-APIs mit Designprinzipien wie einer Client-Server-Architektur, Zustandslosigkeit, Caching, einer einheitlichen Schnittstelle und Schichtenarchitektur ein, erläutert die Architektur des ABAP RESTful Application Programming Model und stellt SAP Fiori Elements als Framework für das einfache Entwickeln einheitlicher, ansprechender Apps vor.
Der praktische Teil untersucht drei Ansätze für die Abbildung asynchroner Prozesse mit sequentieller Kommunikation: Business Workflows, Business Events und das Background Processing Framework (bgPF). Workflows ermöglichen die Abbildung verschiedener Geschäftsprozesse im SAP-System und eignen sich für repetitive Prozesse. Business Events ermöglichen asynchrone Kommunikation zwischen Business Objects, während das bgPF die Auslagerung von Logik in separate Prozesse erlaubt.

Die Vergleichsanalyse zeigt, dass Workflows potenziell komplex sind, während Business Events je nach Ansatz variieren und das bgPF eher einfach einzusetzen ist. In Bezug auf die Systemlandschaft benötigen Workflows keine zusätzlichen Anforderungen, während Business Events und bgPF spezifische Broker bzw. Event Meshes erfordern. Workflows zeigen gute Performance, jedoch können Benutzerinteraktionen die Ausführung blockieren. Business Events sind effizient, aber die Größe der Daten kann die Performance beeinflussen, während das bgPF durch seine asynchrone Natur eine gewisse Verzögerung aufweist. Die Kosten variieren bei den Ansätzen: Workflows und das bgPF verursachen keine zusätzlichen Kosten, während Business Events Investitionen erfordern können. Hinsichtlich Flexibilität bieten Workflows und das bgPF Anpassungsmöglichkeiten durch ABAP-Coding, während Business Events ebenfalls anpassbar sind und gut in die Systemlandschaft integriert werden können. In Bezug auf Skalierbarkeit zeigen Workflows nur lineares Wachstum, Business Events sind sehr gut skalierbar, während das bgPF hier derzeit noch Defizite im Bezug

auf die Skalierbarkeit hat und ausschließlich in Cloud-Systemen verfügbar ist. Die Wartbarkeit von Workflows und dem bgPF ist relativ einfach, während Business Events von der korrekten Implementierung abhängig sind, aber gut gewartet werden können. Abschließend wird die Abwärtskompatibilität betrachtet: Workflows sind sehr gut abwärtskompatibel, während Business Events und bgPF von der verwendeten Technologie und Version abhängen. Workflows bieten sich somit an, wenn Abwärtskompatibilität wichtig ist.

In der Entscheidungsmatrix zeigt sich, dass Workflows breite Stärken haben, Business Events besonders skalierbar und effizient sind, während das bgPF Framework besonders für Cloud-Systeme geeignet ist. Alle drei Ansätze sind je nach spezifischem Anwendungsszenario nutzbar.

4.2. Handlungsempfehlung

Abschließend soll eine Handlungsempfehlung für den konkreten Anwendungsfall in der Abteilung gegeben werden. Die nachfolgende Tabelle stellt eine gewichtete Entscheidungsmatrix der Kriterien, anhand derer die verschiedenen Ansätze verglichen wurden, dar.

		Business Workflows		Business Events		bgPF	
Entscheidungskriterien	**Gewichtung (100)**	**Wertung (1-5)**	**Punktzahl**	**Wertung (1-5)**	**Punktzahl**	**Wertung (1-5)**	**Punktzahl**
Komplexität der Implementierung	20	2	40	3	60	4	80
Auswirkungen auf die Systemlandschaft	10	5	50	1	10	5	50
Performance	15	4	60	4	60	3	45
Kosten	5	5	25	2	10	5	25
Flexibilität	20	4	80	4	80	3	60
Skalierbarkeit	5	1	5	5	25	2	10
Wartbarkeit	20	4	80	3	60	4	80
Abwärtskompatibilität	5	5	25	3	15	2	10
Summe	100		365 /500		320 /500		360 /500

Tabelle 1.: Gewichtete Entscheidungsmatrix der drei Ansätze. Eigene Darstellung

Die Gewichtungen der Kriterien wurden durch eine abteilungsinterne Befragung der für das Thema verantwortlichen Personen ermittelt. Die Wertungen der Technologien für die einzelnen Vergleichspunkte wurden aus dem Vergleich im vorhergehenden Kapitel bestimmt. Somit ergibt sich, dass die Technologie Business Workflows, die auch aktuell zum Lösen der Problemstellung im Betrieb zum Einsatz kommt tatsächlich, entgegen der anfänglichen Annahme die beste Variante ist, wenn auch der Unterschied zum bgPF nur sehr marginal ist (Vgl. Abb. 6). Da die Bewertungen relativ zueinander erfolgt sind, bei den beiden Technologien nur um 1% abweichen

und generell ähnliche Stärken haben, sollten diese als gleichwertig geeignet für die Lösung der Problemstellung betrachtet werden. Business Events würden sich zwar grundlegend auch für den Anwendungsfall eigenen, bieten aber insgesamt nicht so viele Vorteile wie die anderen beiden Ansätze. Allgemein hängt die Wahl von der spezifischen Gewichtung der einzelnen Kriterien ab.

4.3. Reflexion der Arbeit und Ausblick

Die anfängliche Fragestellung der Arbeit war, welcher der drei vorgestellten Lösungsansätze am besten im betrieblichen Anwendungsszenario eingesetzt werden sollte und welcher Ansatz sich allgemein anhand mehrerer Vergleichskriterien wofür eignet. Im letzten Kapitel des Praxisteils und im vorhergehenden Kapitel der Arbeit wurden diese Fragen nun beantwortet. Es wurde gezeigt welcher Ansatz für die Abteilung der geeignetste wäre sowie allgemein die Stärken und Schwächen der einzelnen Ansätze herausgestellt, sodass die Ergebnisse ohne Weiteres auch auf andere ähnliche Szenarien übertragen werden können. Möglichkeiten der weitergehenden Forschung bestehen vor allem darin, die vorgestellten Ansätze anhand von Prototypen oder Proof-of-Concepts in der Praxis anzuwenden und somit auch eine praktische Referenz für die Umsetzung sequentieller Prozesse im RESTful-API Umfeld mit einem transaktionalen Kontext zu bilden. Des Weiteren können auch noch weitere Ansätze zur Umsetzung solcher Prozesse betrachtet werden, die in dieser Arbeit aufgrund des Umfangs keine Berücksichtigung finden konnten.

Literaturverzeichnis

[1] SAP. *Geschichte der SAP | Über SAP SE*. German. 2023. URL: https://www.sap.com/germany/about/company/history.html (Einsichtnahme: 13. 07. 2023) (siehe S. 1).

[2] Biehl, M. *API Architecture*. API-University Series. CreateSpace Independent Publishing Platform, 2015. URL: https://books.google.de/books?id=6D64DwAAQBAJ (Einsichtnahme: 18. 07. 2023) (siehe S. 4).

[3] Fielding, R. „Architectural Styles and the Design of Network-based Software Architectures“. Diss. Irvine: University of California, 2000 (siehe S. 4–7).

[4] SAP. *ABAP RESTful Application Programming Model | SAP Help Portal*. English. 2023. URL: https://help.sap.com/docs/btp/sap-abap-restful-application-programming-model/abap-restful-application-programming-model?locale=en-US (Einsichtnahme: 13. 07. 2023) (siehe S. 8–12).

[5] SAP. *SAP UI5 Documentation, Developing Apps with Fiori Elements*. English. 2022. URL: https://ui5.sap.com//#/topic/03265b0408e2432c9571d6b3feb6b1fd (Einsichtnahme: 13. 07. 2023) (siehe S. 12, 13).

[6] SAP. *Developing SAP Fiori Applications with SAP Fiori Tools | SAP Help Portal*. English. 2023. URL: https://help.sap.com/docs/SAP_FIORI_tools/17d50220bcd848aa854c9c182d65b699/f09752ebcf63473e9194ea29ca232e56.html (Einsichtnahme: 13. 07. 2023) (siehe S. 13).

[7] SAP. *SAP Business Workflow | SAP Help Portal*. English. 2022. URL: https://help.sap.com/docs/SAP_NETWEAVER_700/109b51f56c531014b4e7c143c4b731a9/4f41e8a0dd89535fe10000000a421937.html?locale=en-US (Einsichtnahme: 17. 07. 2023) (siehe S. 14–16).

[8] SAP. *Business Events | SAP Help Portal*. English. 2023. URL: https://help.sap.com/docs/btp/sap-abap-restful-application-programming-model/business-events (Einsichtnahme: 17. 07. 2023) (siehe S. 16, 17).

[9] SAP. *Creating an Event Consumption Model | SAP Help Portal*. English. 2022. URL: https://help.sap.com/docs/btp/sap-abap-development-user-guide/creating-event-consumption-model?locale=en-US (Einsichtnahme: 17. 07. 2023) (siehe S. 17).

A. Anhang

A.1. Fragebögen Experteninterviews

A.1.1. Workflows

Allgemein

Wer bist du und was ist deine Aufgabe in der AIS? Was machst du in deinem täglichen Alltag und wo kommst du mit Workflows in Kontakt? Darf ich das Interview aufzeichnen, transkribieren und in meiner Praxisarbeit verwenden?

Komplexität der Implementierung

Wie komplex ist eine Implementierung von Workflows? (genauere Beschreibung der Implementierung, welche/ wie viele Artefakte müssen erstellt werden, ca. Angabe Lines of Code)

Auswirkungen auf die Systemlandschaft

Hat die Implementierung von Workflows Auswirkungen auf die Systemlandschaft? (Zusätzliche Systemkomponenten, Cloud (weitere Implikationen: Datenschutz, rechtliches,) nötig,)

Performance

Wie sind Workflows performance-technisch einzuordnen? (Rechenlast, Ausführungszeit, Netzwerklast)
Ist für Workflows eine Kommunikation über Systemgrenzen hinaus notwendig?

Kosten

Entstehen durch die Implementierung von Workflows zusätzliche Kosten? (Lizenzkosten, Cloud-Abonnement (laufende Kosten), Netzwerk-Traffic)

Flexibilität

Wie flexibel sind Workflows im Bezug auf ihre spätere Anpassbarkeit, Gestaltungsmöglichkeiten bei speziellen Anforderungen, Integrationsmöglichkeiten mit anderen Technologien?

Skalierbarkeit

Wie skalierbar sind Workflows? (Aufteilbar (Load-Balancing) auf mehrere Systeme, Frameworks die Skalierung übernehmen) Bezug auf Fiori-Apps mit sehr hohem Nutzungsvolumen

Wartbarkeit

Beschreiben Sie die Wartbarkeit von Workflows (Wartung zentral über mehrere Instanzen verteilt, Analysemöglichkeiten).

Abwärtskompatibilität

Sind Workflows abwärtskompatibel zu älteren Releases bzw. älteren SAP-Systemen?

A.1.2. Business Events

Allgemein

Wer bist du und was machst du bei SAP?
Wo kommst du mit Business Events in Kontakt?
Darf ich das Interview aufzeichnen, transkribieren und in meiner Praxisarbeit verwenden?

Komplexität der Implementierung

Wie komplex ist eine Implementierung von Business Events? (genauere Beschreibung der Implementierung, welche/ wie viele Artefakte müssen erstellt werden)

Auswirkungen auf die Systemlandschaft

Hat die Implementierung von Business Events Auswirkungen auf die Systemlandschaft? (Zusätzliche Systemkomponenten, Umstrukturierung von Prozessen, Migrationsaufwand on-premise Landschaft zu eventgesteuerter Architektur, Cloud nötig,)

Performance

Wie sind Business Events performance-technisch einzuordnen? (Rechenlast, Ausführungszeit, Netzwerklast)
Ist für Business Events eine Kommunikation über Systemgrenzen hinaus notwendig?

Kosten

Entstehen durch die Implementierung von Business Events zusätzliche Kosten? (Lizenzkosten, Cloud-Abonnement, Netzwerk-Traffic)
In welchem Verhältnis stehen diese zum Mehrwert, der durch eine eventgesteuerte Architektur entsteht?

Flexibilität

Wie flexibel sind Business Events im Bezug auf ihre spätere Anpassbarkeit, Gestaltungsmöglichkeiten bei speziellen Anforderungen, Integrationsmöglichkeiten mit anderen Technologien?

Skalierbarkeit

Wie skalierbar sind Business Events? (Aufteilbar (Load-Balancing) auf mehrere Systeme, Frameworks die Skalierung übernehmen) -> Bezug auf Fiori-Apps mit sehr hohem Nutzungsvolumen

Wartbarkeit

Beschreiben Sie die Wartbarkeit von Business Events (Wartung zentral über mehrere Instanzen verteilt, Analysemöglichkeiten).

Abwärtskompatibilität

Sind Business Events abwärtskompatibel zu älteren Releases bzw. älteren SAP-Systemen?

A.1.3. bgPF

Allgemein

Who are you and what do you do at SAP?
Where do you encounter bgRFCs or the bgPF in your daily work?
May I record the interview, transcribe it and use it in my practice work?

Komplexität der Implementierung

How complex is an implementation of a bgRFC/ bgPF (more detailed description of the implementation, what/how many artifacts need to be created). 7x

Auswirkungen auf die Systemlandschaft

Does the implementation of bgRFC/ bgPF have any impact on the system landscape? (Additional system components, restructuring of processes, cloud necessary, ...)

Performance

How should bgRFC/ bgPF be classified in terms of performance? (Computing load, execution time, network load) Is communication across system boundaries necessary for business events?

Kosten

Does the implementation of bgRFC/ bgPF result in additional costs? (License costs, cloud subscription, network traffic)

Flexibilität

How flexible are bgRFCs/ bgPF in terms of their subsequent adaptability, design options for special requirements, integration options with other technologies?

Skalierbarkeit

How scalable are bgRFCs/ bgPF? (Divisible (load-balancing) to multiple systems, frameworks that handle scaling) Reference to Fiori apps with very high usage volume.

Wartbarkeit

Describe the maintainability of bgRFC/ bgPF. (Maintenance centralized or distributed across multiple instances, analytics capabilities).

Abwärtskompatibilität

Are bgRFC/ bgPF backward compatible with older releases or older SAP systems?

A.2. Transkripte Exptertenineterviews

A.2.1. Business Workflows

Befragender: Tom Wolfrum (Abkürzung: **T**)

Befragter: Eric Serie (Abkürzung: **E**)

Datum: 21.07.2023

T: Hallo Eric, vielen Dank, dass du dir die Zeit für dieses Interview im Rahmen meiner Praxisarbeit genommen hast. Thematisch soll es heute um die SAP-Technologie

Business Workflows gehen. Doch starten wir bei dir als Person. Wer bist du und was ist deine Aufgabe in unserer Abteilung AIS HCM? Du kannst auch darauf eingehen, was du in deinem Arbeitsalltag machst und wo du mit Workflows in Kontakt kommst.

E: Hallo Tom, ich bin Eric Serie. Ich habe 1997 bei SAP angefangen. Ich war anfangs in einer französischen Abteilung und jetzt seit über zehn Jahren in der AIS HCM. In meinem Arbeitsalltag kümmere ich mich größtenteils um Kundenmeldungen und helfe Kollegen bei Fragen. Mein Bereich ist die Personaladministration und ich betreue die Workflows der SAP Basis. Hier komme ich mit Workflows in Kontakt. Ich betreue den Teil der Workflows, der mit Personalentwicklung und Objekten wie Planstellen, Organisationseinheiten und Personalnummern zu tun Hauptprozess.

T: Danke für deine kurze Vorstellung. Noch vorneweg: Darf ich das Interview aufzeichnen, transkribieren und - wenn du damit einverstanden bist - in meiner Praxisarbeit verwenden?

E: Ja.

T: Ok, super. Dann kommen wir zu den inhaltlichen Fragen. Zur Komplexität der Implementierung: Kannst du beschreiben, wie komplex eine Implementierung von Workflows ist, also was man genau machen muss und welche Artefakte erstellt werden müssen?

E: Ein Workflow kann sehr einfach sein, also eine einfache Aufgabe erfüllen, wie z. B. eine E-Mail versenden oder sehr komplex, da ein Workflow viele Aufgaben oder Schritte haben kann und diese können dann auch wieder sehr verschieden sein, wie zum Beispiel eine Entscheidung oder Genehmigung oder eine komplizierte Aufgabe, wie eine Abwesenheit anzulegen. Es gibt bei Workflows viele verschiedene Schritte und Schritttypen, deshalb können sie sehr komplex sein. Zudem kann in einem Schritt eines Workflows eigenes ABAP-Coding ausgeführt werden, was Workflows auch nochmal komplexer macht.

T: Okay, dass heißt, dass die Komplexität eines Workflows sozusagen mit der Anzahl seiner Schritte steigt?

E: Ja, das ist auf jeden Fall so.

T: Kommen wir zur nächsten Frage: Hat die Implementierung von Workflows Auswirkungen auf die Systemlandschaft, also braucht man zusätzliche Systemkomponenten, sind Cloud-Komponenten nötig?

E: Nein, Workflows sind komplett in der SAP-Basis enthalten und jedes System hat diese Funktionalität.

T: Das heißt, dass wenn sich der Kunde ein SAP ERP-System kauft, ist das dort alles mit enthalten?

E: Genau.

T: Meine nächste Frage betrifft den Punkt Performance: Wie würdest du sagen, sind Workflows performance-technisch einzuordnen? Mögliche Kriterien wären hier Rechenlast, Ausführungszeit oder Netzwerklast.

E: Die Performance ist bei Workflows kein Problem, die Ausführung geht sehr schnell.

T: Ok, das heißt, dass das auch im Bezug auf Fiori-Apps, wie zum Beispiel der Prozess einer Krankmeldung, bei denen der Workflow sehr häufig gestartet würde, kein Problem wäre?

E: Ja, das wäre kein Problem, weil die Prozesse hinter Workflows meist so aufgebaut sind, dass immer nur kleine Schritte ausgeführt werden und der Workflow nicht auf einmal durchläuft. Zudem sind die hintereinander ausgeführten Schritte meist sehr verschieden und sind für sich sehr schnell in der Ausführung. Häufig sind zum Beispiel Benutzerentscheidungen notwendig, wo auf den Anwender gewartet werden muss und somit ist die Performance der Workflows an sich eigentlich nie ein Problem. Der einzige Fall, in dem das nicht so ist, wäre, wenn ein Fehler in der Anwendung passiert, was natürlich nur ein Ausnahmefall ist.

T: Ok, das heißt, dass Workflows an sich performance-technisch sehr gut sind, weil meist nie der ganze Prozess bzw. Workflow auf einmal durchläuft, sondern eben nur ein kleiner Schritt und die auch nicht alle auf einmal?

E: Genau, aber selbst wenn mehrere Schritte auf einmal ausgeführt werden, ist das auch kein Problem, da das alles im Hintergrund passiert und für den Anwender nicht sichtbar ist.

T: Und findet die Ausführung von Workflows komplett lokal statt oder ist hier die Kommunikation über Systemgrenzen hinaus notwendig?

E: Es kann sein, dass die Kommunikation mit anderen Systemgrenzen durch den abgebildeten Prozess notwendig ist, aber das ist von der Performance her kein Problem. Zudem findet der Großteil der Ausführung von Workflows lokal statt.

T: Damit einhergehend der Kostenfaktor: Entstehen durch die Implementierung von Workflows irgendwelche zusätzlichen Kosten?

E: Nein, gar nicht.

T: Ok, dann auch hier alles mit dem einmaligen Kauf der SAP-Softwarelizenz abgedeckt?

E: Ja.

T: Dann kommen wir zur Flexibilität von Workflows: Wie flexibel sind Workflows im Bezug auf ihre spätere Anpassbarkeit, Gestaltungsmöglichkeiten bei speziellen Anforderungen und auch auf Integrationsmöglichkeiten mit anderen Technologien?

E: Workflows sind ziemlich flexibel, SAP liefert zwei Arten von Workflows aus: klassische und flexible Workflows. Beim klassischen Workflow liefert SAP große Muster-Workflows für bestimmte Geschäftsprozesse aus und der Kunde kopiert sich diese Muster und passt diese dann auf seine Bedürfnisse an. Klar, der Kunde muss etwas von der Technik verstehen, aber er muss die Muster nur an seine Gegebenheiten anpassen. Der Nachteil an diesem Konzept ist, dass wenn der Muster-Workflow fehlerhaft ist und wir eine Korrektur veröffentlichen, diese Korrektur nicht automatisch im System des Kunden übernommen wird, sondern der Kunde seinen Workflow selbstständig anpassen muss. Für die Cloud gibt es ein neues Konzept, die flexiblen Workflows. Hier liefern wir nur kleinere einzelne Schritte bzw. Workflows, die der Kunde dann in seinen Workflow integriert. Sollte dann ein Fehler in diesen kleinen Schritten gefunden werden, ist nur dieser kleine Teil des Workflows betroffen und der Kunde bekommt zudem automatisch die Korrektur. Zudem sind diese kleineren Workflows flexibler, da sie sich modular zusammensetzen lassen.

T: Ok, ich fasse das nochmal zusammen: Vorher [Anmerkung: on-premise Umfeld] war es so, dass SAP einen großen Workflow ausgeliefert, der große Prozesse abgedeckt hat und den der Kunde nur noch an sich anpassen musste. Wenn in diesem Workflow ein Fehler gefunden wurde, musste der Kunde, euere Korrektur trotzdem noch einmal selbst bei sich anpassen. Jetzt ist dann sozusagen das Ziel, dass nur noch kleinere Workflows ausgeliefert werden.

E: Ja genau, jetzt liefern wir nur noch kleinere Teile und aus diesem muss der Kunde nur noch seinen Workflow zusammensetzen. Wenn SAP einen Fehler in diesem kleinen Teilen findet, bekommt der Kunde die Korrektur automatisch.

T: Du hast ja gesagt, dass sich Workflows aus einzelnen Schritten zusammensetzen. Das heißt, dass der Kunde sich auch aus allen verfügbaren Einzelschritten bei einer speziellen Anforderung seinen komplett eigenen Prozess frei zusammenbauen, also er ist nicht an die SAP-Vorlagen gebunden?

E: Ja genau, da hast du recht.

T: Kannst du etwas zu Integrationsmöglichkeiten von Workflows zu anderen Technologien sagen. Du hast beispielsweise schon das Versenden von E-Mails angesprochen.

E: Da Workflows Teil der SAP-Basis Softwarekomponente sind, können diese mit allen anderen Komponenten/ Schichten interagieren. Somit kann jeder Entwickler auf die Workflows in der SAP-Basis zugreifen und diese gegebenenfalls an ihre Bedürfnisse anpassen. Deshalb sind Workflows gut mit anderen Technologien integrierbar.

T: Hier spielt dann ja auch wahrscheinlich mit hinein, dass man in einem Workflow-Schritt sein eigenes ABAP-Coding ausführen kann und somit alles machen kann?

E: Ein Beispiel hierfür wären Berechtigungen: Ein Basis-Workflow kann keine Berechtigungen, die eventuell in anderen Anwendungen benötigt werden, prüfen. Hier kann man beispielsweise durch eigenes ABAP-Coding diese Berechtigungsprüfungen selbst implementieren und die Funktionalität des Workflows beliebig erweitern.

T: Ok, danke. Dann zur nächsten Frage: Wie skalierbar sind Workflows? Ist die Ausführung eines Workflows auf mehrere Prozesse aufteilbar und gibt es Frameworks, die die effiziente Skalierung übernehmen? Gerade wir in der AIS setzen Workflows ja auch in Fiori-Apps ein, die ja ein sehr hohes Nutzungsvolumen haben.

E: Das ist eine gute Frage. Ich würde sagen, dass Workflows an sich sehr flexibel sind. Die Workflows der Basis sind sehr performant und schnell in der Ausführung. Normalerweise gibt es hier keine Probleme bei Anwendungen, die viele Workflows ausführen müssen.

T: Das heißt der Workflow an sich ist ein großer zusammenhängender Prozess, der nicht nochmal unterteilt und über mehrere Applikationsserver verteilt ausgeführt werden kann.

E: Ja genau, da hast du recht.

T: Zum Thema Wartbarkeit: Wie wartbar sind Workflows? Also kann ich das zentral an einer Stelle machen oder muss ich Probleme über mehrere Komponenten verteilt suchen?

E: Nein, die Workflows werden an einer Stelle erstellt und können auch von dort aus zentral gewartet werden, auch wenn sie andere Komponenten beeinflussen. Die Wartung ist ziemlich einfach.

T: Dann kommen wir auch schon zu meiner letzten Frage, die Abwärtskompatibilität. Oft findet man die Situation vor, dass Kunden eine Systemlandschaft aus neueren und älteren Systemen haben. Dahingehend die Frage, wie abwärtskompatibel sind Workflows?

E: Klassische Workflows funktionieren in allen Systemen. Flexible Workflows jedoch nicht, da es diese noch nicht so lange gibt und nur für die Cloud gedacht sind. Klar gibt es im Bezug auf die UI einige Verbesserungen mit den neuren Versionen, aber abgesehen davon ist es in allen Systemen das Gleiche. Auch das Entwickeln eines Workflows in einem höheren Release und das Herunterziehen in einen älteren Release ist überhaupt kein Problem.

T: Ok gut, das wars dann auch schon mit meinen Fragen. Dann bedanke ich mich für das Interview und würde dir dann das Transkript nochmal zuschicken, damit das für dich passt und ich das dann in meiner Arbeit verwenden kann.

E: Ok, so machen wir das.

A.2.2. Business Events

Befragender: Tom Wolfrum (Abkürzung: **T**)

Befragter: Karsten Strothmann (Abkürzung: **K**)

Datum: 24.07.2023

T: Hallo Karsten, vielen Dank, dass du dir die Zeit für dieses Interview im Rahmen meiner Praxisarbeit genommen hast. Thematisch soll es heute um die SAP-Technologie Business Events gehen. Doch starten wir bei dir als Person. Wer bist du und was machst du bei SAP? Du kannst auch darauf eingehen, wo du in deinem täglichen Alltag mit Business Events in Kontakt kommst.

K: Mein Name ist Karsten Strothmann, ich bin Lead Product Manager für das Event Mesh und SAP Integration Suite Advanced Event Mesh, dass heißt eigentlich für die Infrastrukturkomponenten, was eventgetriebene Architekturen angeht. Dabei bleibt es aber üblicherweise nicht, weil letztenendes müssen die Events irgend-

woher kommen, dass heißt Business Applications, Back-End-Systeme und die müssen natürlich irgendwo konsumiert werden, dass heißt üblicherweise habe ich die End-to-End-Brille auf und schaue da so ein bisschen hollistisch drauf, das heißt verschiedenste Themen und Perspektiven. Ich habe eine recht hohe Bandbreite und die muss ich auch haben, weil das Thema immernoch recht neu ist und es noch sehr viele Themen, wie Rechtliches, Coding, Roll-out/ -in bei Kunden und Definitionen von komplett neuen Prozessen zu klären gibt. Vieles ist einfach noch komplettes Neuland und ich arbeite sehr viel mit Kunden und auch bei den Kunden ist es so, dass sie gerade erst anfangen. Wie bin ich dazu gekommen? Ich habe vor einigen Jahren mal an einem Produkt namens SAP Gateway gearbeitet, was dazu dient synchron SAP Back-End-Systeme zu erweitern, das heißt, man hat einen OData-Call, kann den konfigurieren, kann teilweise entwickeln um die Back-End-Systeme zu öffnen und ruft diese über APIs synchron auf. Das hat sich in Richtung asynchrone Kommunikation entwickelt. Ansonsten bin ich fast 25 Jahre bei der SAP und habe davor bei einem Beratungshaus gearbeitet.

T: Dann noch formal: Darf ich das Interview aufzeichnen, transkribieren und in meiner Praxisarbeit verwenden?

K: Darfst du sehr gerne machen.

T: Ok super, ich würde jetzt mit meinen inhaltlichen Fragen zu Kriterien, anhand denen ich Business Events mit anderen Technologien vergleiche, beginnen. Die erste Frage wäre, wie komplex es ist Business Events zu implementieren. Vielleicht kannst du hier etwas genauer beschreiben, wie die Implementierung abläuft und welche bzw. wie viele Artefakte man erstellen muss.

K: Ich fange mal mit der Kundenebene an. Für den Kunden soll es natürlich so einfach wie möglich sein. Idealerweise, wie es bei unseren Standard Events ist, braucht man 10 Sekunden, nachdem die Infrastruktur eingerichtet ist, um ein neues Event aktiv zu schalten, das heißt, die Events können durch das einfach Umlegen eines Schalters exponiert werden. Ganz wichtig ist es, dass die Events aktiv im Back-End-System eingeschaltet werden mmüssen und nicht standardmäßig eingeschaltet sind. Die Events werden erstellt bzw. entwickelt von unserer Entwicklung, das heißt die erstellen auch die zugehörigen Artefakte, auf die du von der Entwicklungsseite hinaus willst. Zudem können Kunden auch noch Custom Events erstellen. Hier hängt der Aufwand von den verwendeten Technologien ab. Eine Möglichkeit sind RAP-basierte Events. Diese sind aktuell noch aufwändiger und man muss RAP-Know-

How haben. Es gibt noch einen zweiten Ansatz, das ist das sogenannte Netweaver Event Enablement Add-On. Das funktioniert mit unseren on-premise Back-End-Systemen, also ECC und S/4. Hier ist es so, dass ich verschiedenste Trigger wählen kann und dann in einem Low-/ No-Code-Ansatz mein Event zusammenklicken kann. So etwas geht normalerweise in 20 Minuten. Was man dafür braucht hängt normalerweise vom Ansatz ab, man kann auch das komplett high-level machen, also konfigurationsbasiert im SAP-UI oder man fängt an sich das Event selbst zu programmieren. Ich selbst habe das noch nicht gemacht, das wäre eher für sehr spezielle Fälle, in denen man Events anpassen will oder auf Back-End-Seite nochmal filtern will, dann kann man das hiermit umsetzen. Das heißt, letztenendes ist der Aufwand für den Kunden komplett unterschliedlich. Von minimal in ca. einer Minute bei Standard-Events über 20 Minuten, um mit dem Event Add-On ein Custom Event zu erstellen bis hin zu drei Tagen für ein hochoptimiertes Event mit RAP. Auch die Artefakte sind je nach verwendeter Technologie und Trigger jedes Mal unterschiedlich. Bei dem RAP-basiertem Ansatz als vierte Variante werden die Standard Events im Herbst eine konfigurierbare Erweiterbarkeit bekommen, das Ganze nennt sich dann "Derived Events". Dadurch kann man die Standard Events dann konfigurationsbasiert erweitern. Die Kernaussage ist, dass das ganze Spektrum vom einfachen Umlegen eines Schalters, ohne das Erstellen von Artefakten bis hin zu einer Reihe von Artefakten bei selbst geschriebenen Events vorhanden ist.

T: Ok, danke. Ein weiteres Kriterium, was ich mir anschaue sind die Auswirkungen auf die Systemlandschaft. Inwiefern hat die Implementierung von Busines Events Auswirkungen auf die Systemlandschaft, also im Bezug auf zusätzliche Systemkomponenten, die Umstrukturierung von Prozessen und der Migrationsaufwand einer reinen on-premise Landschaft zu einer eventgesteuerten Architektur mit Business Events?

K: Also letztenendes muss man natürlich seine Prozesse anpassen, da man von synchronen batchorientierten Prozessen weg, hin zu komplett asynchronen, hoch paralleln Prozessen umsteigt. Ein passender Vergleich wäre der eines Nachrichtendienstes: Im Back-End passiert etwas, das Event wird abgeschickt und jeder der auf ein Event lauscht, wird informiert über die Änderung. Man hat sozusagen eine geschlossene Gruppe in die ein Event geschickt wird und über das dann jeder in dieser Gruppe informiert wird. Wenn man dann sozusagen vom Telefonieren als synchrone Kommunikation zu einem Nachrichtendienst als asynchrone Kommunikation wechselt, stellt man optimalerweise auch seinen Geschäftsprozess um. Das heißt, es wird

definitiv Änderungen dabei geben. Sehr viele Kunden haben bei dieser Umstellung ein Problem, da man versucht mit alten Sichtweisen an die neuen Ansätze heranzugehen, was für Probleme sorgt. Wenn man von einem batch-basiertem Ansatz der einmal am Tag läuft zu einem Echtzeit-Ansatz wechsele, müssen die Geschäftsprozesse zwangsläufig angepasst werden.

T: Ganz allgemein: Wenn ich Business Events benutzen möchte, brauche ich aber schon eine Cloud-Komponente in meiner Systemlandschaft, wie eben das Event-Mesh?

K: Jaein, es wird ein Event Broker oder Mesh benötigt. Der SAP Event Mesh ist eigentlich nur ein Event Broker, der irreführend bennant ist. Man braucht mindestens ein Broker, wie zum Beispiel das SAP Event Mesh als Cloud Komponente. Wir positionieren zusätzlich gerade den Advanced Event Mesh, womit man tatsächlich ein Netzwerk aus mehreren Brokern erstellen kann. Das ist zur Zeit die Marschrichtung. Das hat den Vorteil, dass unsere Kunden meistens global aufgestellt sind und ihre Back-End-Systeme weltweit verteilt sind. Meistens wird ein Event Broker neben das Back-End-System gestellt, dem das Event übergeben wird und müssen sich dann um nichts mehr kümmern. Der weitere Punkt ist, dass diese Event Broker nicht in der Cloud stehen. Das ist vor Allem für Kunden, die nicht mit ihren Daten in die Cloud gehen möchten, von Vorteil, da man diesen Event Broker auch lokal on-premise betreiben kann. Insgesamt ist die Cloud somit keine zwingende Voraussetzungen, wenn sie auch enorme Vorteile, wie die globale Vernetzung der Systeme mit Events bietet. Um es ganz deutlich zu sagen: Es wird ein Event Broker als zusätzliche Systemkomponente benötigt. In den ABAP-Systemen gibt es schon sehr lange Events. Diese sind jedoch immer nur lokal genutzt worden. Jetzt ist der große Unterschied, dass diese Events jetzt extern und global verteilt werden können, teilweise über Herstellergrenzen hinweg.

T: Das heißt, dass ich mit dem SAP Event Mesh auch Events anderer Hersteller verarbeiten könnte?

K: Unsere Events sind normalerweise im Cloud Events Format, das von Microsoft, Google, SAP, Oracle und weiteren Tech-Konzernen vorangetrieben wird. Das heißt, wir können die Events auch unter den Herstellern austauschen.

T: Danke. Mein nächstes Kriterium wäre die Performance. Wie kann man generell Business Events performancetechnisch einordnen? Mögliche Messgrößen wären hier Rechenlast, Ausführungszeit und Netzwerklast.

K: Die Events können unterschiedlich groß sein, von extrem kleinen Notification-Events bis hin zu sehr großen Daten-Events. Lasst uns das mal versuchen zu strukturieren: Das Back-End-System, der Event Broker und die Konsumenten. Für das Back-End und die Konsumenten ist es von enormen Vorteil, da die Alternative zum eventgesteurtem Ansatz das permanente Rufen und Prüfen auf Änderungen des Back-End-Systems ist. Dadurch, dass dieses permanente Rufen entfällt wird erheblich Performance gespart, da einfach nur ein Event losgeschickt wird, wenn tatsächlich etwas passiert. Der Broker ist darauf ausgelegt, enorm zu skalieren. Bei der Erstellung von diesen Events muss man sehr vorsichtig sein, dass man nur wirklich benötigte Events erstellt und dass die Events möglichst groß werden. Denn in diesem Moment erezugt man Last auf dem Back-End-Sytem. Man muss die Balance zwischen dem Einsparen von API-Calls und dem Erzeugen der Events in der richtigen Größe. Hier ist definitiv ein großes Optimierungspotenzial bei den S/4-Kollegen, gerade weil unsere Back-End-System etwas dahin tendieren, sehr monilithisch zu sein.

T: Das ging jetzt vielleicht schon etwas aus den vorherigen Fragen hervor, aber die Kommunikation über Systemgrenzen hinaus liegt ja in der Natur der Business Events, oder?

K: Richtig, Busienss Events sind darauf ausgelegt. Die Standardvorgehensweise ist meistens, dass die SAP-Events aus den Systemen nach außen geschickt werden, da die SAP-Back-End-Systeme als Eventquellen sehr gut sind und die SAP-Events sehr wertvoll sind. Dass andere Events von außen in die SAP-Systeme kommen ist etwas seltener, aber passiert definitiv auch, also es soll und muss definitiv in die Richtungd herstellerübergreifende Kommunikation gehen.

T: Ok, mein nächster Punkt wäre die Kostenseite, also entstehen durch die Implementierung von Business Events zusätzliche Kosten, wie zum Beispiel durch Cloud Komponenten und in welchem Verhältnis stehen diese zum Mehrwert, den eine eventgesteuerte Architektur bietet?

K: Das sind ziemlich gute Fragen, die auch in der Forschung noch nicht so richtig beleuchtet wurden. Ein Broker kostet definitiv Geld. Es gibt allerdings unterschiedlichste Größenordnungen. Wenn ich den SAP Event Mesh nehme, ist der ernorm günstig, da er auf die Verwendung von kleinen Notification-Events ausgelegt ist und die Kunden in Abhängigkeit von ihrem Nutzungsvolumen bezahlen. Meistens läuft es hier im Monat auf mehrere Hundert Euro hinaus. Jedoch ist er auch nicht

darauf ausgelegt, ein ganzes Unternehmen zu bedienen. Das heißt, man bedient hier lediglich Teilbereiche und spezielle Anwendungsfälle, dafür sind die Kosten aber auch nur sehr gering. Um die komplette Architektur eines Unternehmens zu bedinenen, egal ob man andere umfassendere Event Broker anderer Hersteller oder den SAP Integration Suite Advanced Event Mesh nimmt, ist eine ganz andere Liga und hier bewegen sich die Beträge eher im sechsstelligen Bereich und aufwärts. Zudem ist der Ansatz ein anderer: Beim Event Mesh findet pay-per-use (Bezahlen pro Benutzung) Anwendung, während man bei solchen größeren Technologien seinen eigenen Broker hat und diesen für die gesamte Zeit, die er läuft bezahlt, egal ob man ihn nutzt oder nicht. Das heißt, da reden wir über große Summen Geld. Es ist keine günstige Technologie, aber es gibt einen Break-even Punkt, ab dem man Geld spart, da die Infrastruktur sowieso da und wiederverwendbar ist. Events zu bauen hingegen ist deutlich einfacher, ich hatte hier vorhin 20 Minuten gesagt, wenn keine Standard Events da sind. In den älteren Ansätzen dauert das deutlich länger und ist wesentlich komplexer. Im Endeffekt reden wir über ca. 100 Events pro Kunde, ab denen der Break-even Punkt überschritten wird. Das kommt immer auf den Anwendungsfall an, es gibt auch Kunden, die den Break-even Punkt schon nach 2 Events erreichen.

T: Ok, dann zum Thema Flexibilität: Wie flexibel sind Business Events im Bezug auf spätere Anpassbarkeit, Gestaltungsmöglichkeiten bei spezielleren Anforderungen und Integration mit anderen Technologien?

K: Erstmal hängt das wieder an den unterschiedlichen Ansätzen, also der verwendeten Technologie. Die Standard Events waren bisher am schlechtesten anpassbar, da konnte man überhaupt nichts machen. Da aber diese Derived Events kommen werden, wird das relativ einfach werden. Diese Custom Events mit dem Add-On waren schon immer sehr flexibel, die können innerhalb kürzester Zeit angepasst werden. Kernaussage ist, dass es von der Eventquelle abhängt, im Regelfall sind die anpassbar, S/4 war bisher die Ausnahme, aber das ändert sich ab Herbst und dann sind auch diese Events sehr flexibel anpassbar. Zur Kombination mit anderen Technologien: Die Events sind zum Teil darauf ausgelegt, dass sie mit anderen Events kombiniert werden, das heißt bei den SAP Standard Events, das waren Notification Events, also sehr klein und enthalten nur die notwendigsten Daten, ist man davon ausgegangen, dass die relevanten Informationen vom Konsumenten sowieso per gesondertem API-Call angefordert werden, wenn das konsumierte Event für ihn interessant ist. Das heißt Events sind gut und einfach kombinierbar,

zumindest solange man sich im SAP-Bereich befindet. Wenn man jedoch eine eventgetriebene Architektur nach der Definition umsetzt, kennt die Eventquelle die Konsumenten der Events nicht. In Kundensystemen ist dies jedoch eigentlich immer der Fall, sodass hier eventgesteuerte Architekturen meist eher zu 80% umgesetzt werden und die Kunden biegen sich die restlichen 20% durch das Treffen von Annahmen an der Eventquelle über den Konsumenten hin.

T: Dann zum Kriterium der Skalierbarkeit: Wie skalierbar sind Business Events?

K: Business Events sind extrem skalierbar. Auch hier wieder die Unterscheidung zwischen Back-End (Quelle) und Broker. Beim Broker reden wir über Milliarden von Events pro Tag, wenn es sein musst. Als Beispiel laufen im Einzelhandel meist eventgetrieben Architekturen und hier kann man sich vorstellen, dass hier enorme Datenmengen beim Abscannen der Artikel über die Systeme laufen. Also das skaliert sehr gut, genau darauf ist es ausgelegt und das ist eine der Stärken.

T: Zwei Fragen habe ich noch: Einmal zum Thema Wartbarkeit: Wie gut wartbar sind Business Events?

K: Business Events sind sehr einfach wartbar, solange man sich an die Grundsätze hält. Standardmäßig wartet man Events nur an der Eventquelle. Wenn man eventgesteuerte Architektur, rein wie in der Lehre umsetzt, da die Events hier sowieso so konstruiert sind, dass sie unabhängig vom Konsumenten sind. In der Realität ist es nicht so ganz einfach, hier muss man etwas an der Eventquelle, am Broker und am Konsumenten machen. Das heißt, je nachdem welcher Ansatz gewählt wird, klassisch eventgetrieben ist sehr einfach wartbar. Angepasst eventgetrieben ist der Aufwand höher, weil potenziell mehrere Stellen angepasst werden, weil z. B. der Konsument Annahmen über die Events schon im Vorhinein trifft. Hier merkt man auch, dass das noch ein neues Thema ist, weil es auf solche Fragen noch keine endgültigen Antworten gibt. Hier reicht das Spektrum von nur kleinen Notfication Events fast ohne Daten, wo man immer einen API-Call machen müsste bis hin zu großen Daten Events die immer gleich alle Daten enthalten und wo dann der Konsument erst entscheiden muss, was er mit dem Event bzw. den Daten macht. Das S/4 ist genau in der Mitte, hier gibt es Decision Events, die genügend Informationen bieten, aber nicht alle. Bei Bedarf können dann noch mehr Informationen per API-Call angefordert werden.

T: Meine letzte Frage betrifft die Abwärtskompatibilität: Gerade im Hinblick auf heterogene Kunden-Systemlandschaften mit teilweise auch älteren Systemen: Wie

abwärtskompatibel sind hier Business Events im Bezug auf ältere Releases oder ältere Systeme?

K: Es kommt wie immer auf die gewählte Variante an. Beim S/4 muss man bei den Standard Events immer die neueste Version haben, da mit den Versionen immer mehr neue Features hinzugefügt werden und es keine Downports gibt. Das ist ein Problem, weil die Kunden immer die neueste Version haben müssen. Bei dem Event Enablement Add-On ist es genau anders herum, da das darauf ausgelegt ist auch mit sehr alten ECC-Systemen zu arbeiten. Man kann Events zwar versionieren, aber dieses Feature kommt auch erst in Zukunft.

T: Gut, das wars dann auch schon mit meine Fragen. Ich bedanke mich für das Interview und würde dir das Transkript dann noch im Nachgang zum drüberlesen zuschicken.

K: Gerne.

A.2.3. bgPF

Befragender: Tom Wolfrum (Abkürzung: **T**)

Befragter: Robert Wang (Abkürzung: **R**)

Datum: 27.07.2023

Anmerkung: Das Experteninterview wurde zum größten Teil über die Technologie bgRFC (background remote function call) geführt. Dies ist jedoch kein Problem und für den Zweck der Arbeit vorteilhaft, da das untersuchte bgPF auf dieser Technologie aufbaut und sich außer in den im Interview angesprochenen Punkten keine Unterschiede ergeben.

T: Hello Robert. Todays interview is about the bgRFC. Thank you for taking the time. I'm just going to start with a few general questions. Who are you and what do you do at SAP? Maybe you can also say something about where you encounter bgRFC in your daily work.

R: My name is Robert, as you know. Im working on the pure ABAP-Stack in the component SAP-Basis, the Netweaver, as it was called and the connectivity team, which belongs to CST Group, which means Client-Server-Technology. I am the code owner of the bgRFC Framework. Basically we maintain and develop new features for the bgRFC Framework and also help other application colleagues to use bgRFC

efficiently. And for my daily work, I would say in 10% of the time I develop new features and the other 90% are allocated to maintenance of the bgRFC. I work with ABAP-Systems, that are shipped to the customer or internally with other applications using bgRFC.

T: Ok, then for the record: May I record, transcribe and use the interview for my project work?

R: Yes, definitely.

T: Ok, thanks. So now the first question about the bgRFC is about the implementation: Can you say something about how complex the implementation of the bgRFC is, maybe describe it with a bit more details or say how many arefacts need to be created. And explain just how it works in general.

R: Yes, sure. If you want to do something based on the bgRFC, you have to do certain things beforehand. For example you have to configure inbound destinations and also tune the configuration for the framework. This is more or less from the on premise system, where the customer has a maintenance team. So for each product, they also have online documentation to tell the customer support team what what steps should be taken to configure bgRFC correctly, and then you can use the bgRFC as technologies to develop your own code. While this part is quite simple, because it is just a syntax where you you call something in the background unit but the function module itself, you have to develop beforehand. So in my opinion is it's quite simple, it's just a couple lines of code.

T: Ok. Then my next question would be about the system landscape of the customer that's implementing it. Does implementing bgRFC impose any kind of implications or impact on the system landscape so e.g.are additional system components required? Do I need to restructure my processes? Do I need cloud components? Or is everything already part of the shipped system and happens there locally?

R: For this question a I would say yes of course. Any new code will impact the bgRFC-calls and but the RFC's will consume the dialogue work processes, so from this point of view, yes. But in bgRFC framework we also have resource control to not screw up the entire system. You can tune the configuration and parameters to either have more resources or less allocated to bgRFC, depending on how quick you you want the bgRFC unit to be processed now, yes I would say yes, but not much of an impact.

T: Ok, but regarding additional system components, because for example another technology that I'm comparing with it needs a a cloud component, so this is not required?

R: Oh ok, no, not really, becaust the bgRFC is on the basis layer of the ABAP System, so its available on every shipped instance and directly ready to use.

T: That's what I meant, so it all happens locally in the system?

R: Exactly, yes.

T: Ok, then my next question is about performance. How can you classify bgRFC in terms of performance? I think, you already mentioned, that you can change, how many ressources it can take up.

R: So first of all, we classify this as asynchronised technology in the ABAP world and it's not a synchronised call, meaning if one application tries to use bgRFC to perform something, they have to expect some delay but it depends on the specific task to be performed. If you e.g. create one million calls in a very short time, so the unit might be running for a long time. So we cannot execute one million calls in for example ten minutes. So this is given and because this is a asynchronous technology, you have to expect some sort of delay but from the pure performance point of view, it depends on the configuration. So you can tune the system to allocate more resources or less resources to be used for bgRFC, so this is very flexible.

T: Ok, so I think the next question might be a pretty short one, but I have to ask the question to compare the technology to others: Does the implementation of bgRFC result in any additional costs for the customer implementing it?

R: No, it's ready in any ABAP System.

T: You already mentioned flexibility. How flexible would you say are bgRFCs in terms of their adaptibility, or if I have special requirements, how flexible is it to meet those requirements and what about integration options with other technologies?

R: bgRFC, as I said is a asynchronised technology, So what we did is basically when you create a bgRFC unit in your own application, the unit will not be called immediately, the call is just saved into the database table. So this is the first step and the second step is, we have a scheduler running all the time and it will then pick up the unit to trigger it. So regarding the integration possibilities: Any application can use

the bgRFC, so there are no scalability problems. We have a scheduler running on each application server. You can basically configure one ore more schedulers and configure when the unit will be triggered. So for example if you have 10 application servers and you really just want have three application servers running the unit, it's also flexible, so you can configure it to not be triggered on the other seven servers.

T: Ok, so my next question would be about scalability. How scalable are bgRFC's? Mayber there is some kind of load balancing across multiple application servers possible? Or the framework is already handling that? I think you already talked about having multiple schedulers.

R: From the bgRFC point, we are only talking about the units. If we have multiple units and multiple queues, the load balancing is done from two perspectives: The first is the scheduler itself and the other one is the execution of the unit. In your ABAP System you can configure for each application server, whether it should run a scheduler or not. So I can configure how many application servers are running a scheduler and how many schedulers are running in one application server. So here it is very flexible. This was the scheduler point of view. When the scheduler triggers a unit, it is also possible to configure on which application server the unit is executed. It can be executed by all aplication servers or only be certain or one server.

T: Ok, the next question would be about maintainability. Can you describe the maintainability of bgRFCs? E.g. is the maintenance possible centrally from one point, or do you have to change something in multiple places?

R: So basically the bgRFC is shipped in default configuration with any ABAP System. If you just configure the destination, then everything is fine. By default, there is one scheduler on each application server and all the application servers belong to the distribution list. For configuration and maintenance, we have a central transaction code, from which you can configure the bgRFC, as we just discussed.

T: Ok, then the next question might also be pretty clear. You just mentioned, that bgRFC is shipped with the basis of the ABAP System. Is bgRFC a technology that already exists for a long time an is therefore downward compatible with older releases or older SAP Systems?

R: Yes, the technology exists since 2005 or 2008, so if we're talking about the ICP Basis release it's compatible down to 701. And you're also talking about the bgPF, if I read your questions right. So the bgPF is a special use of bgRFC and mainly

still being developed. It is meant for the cloud, so for the ABAP Steampunk system, to allow the customer to develop something in the cloud system and to be able to use bgRFC. Because what we just talked about is just for on-premise systems, because in the cloud, bgRFC is not whitelisted, meaning the customers can't use bgRFC there. So they have to use bgPF as a wrapper, like an API to use it.

T: That's what your colleague Bernhard from Germany also told me, that the bgPF is only a wrapper, to make the bgRFC functionality available in ABAP Cloud. Would you say, if you look at the questions I just asked, that there are any differences when using bgPF instead of bgRFC?

R: Technically there are no differences, the only difference is the programm modelling. If you use the bgRFC directly, you are able to develop and call your own function module. If you then use bgRFC, it will trigger your function module. But in the bgPF, everything is object oriented and you don't need to develop your own function module. You can create a bgPF unit and basically the technology behind it is just a serializer to serialize your object and save it to the database table. There also is a special fixed function module behind the scenes, which the developer won't see. They use this function module to transport the serialized object to the target and in the target they just deserialize it again to initialize a new object to do your logic. So this is a difference, but technically it's still the function module being called by the bgRFC framework.

T: Ok, but looking at the criterias like e.g. flexibility or scalability, does it make a difference if I use one technology or the other?

R: If we are talking about flexibility or scalability, bgPF is a little bit different, because it's a specialized usage of the bgRFC. The bgPF can only use one destination, which makes a difference in ressource allocation, which is configured on a destination basis. If you have for example many bgPF units, everything still goes to one destination, whereas with bgRFC you can define different destinations, and equally move your units there. This means, that the bgRFC is more flexible and scalabe in that point. If you choose the bgRFC and create multiple destinations, you can allocate more ressources to them, wich will result in better performace. Maybe in the future this will also be possible with bgPF.

T: That's what also Bernhard that the bgPF is a bit limited right now and can't do everything that the bgRFC can do at its current state. If we talk about how complex

the implementation is, are there any differences, when using one technology or the other?

R: Yes, so the main difference is, that the bgRFC is not exposed in the cloud system, so there bgPF is the only option. From a technical point of view, as I said, bgPF is only a specialized usage of the bgRFC. Mayber in the future, the bgPF could also be made available for on-premise Systems.

T: Yes, Bernhard told me that it will be made available for on-premise Systems. I don't recall the exact release, but I think it was the 2023 version.

R: Yes, so basically for Steampunk and on-premise, we share the same code for the bgPF in my layer. So if everything is ready for the cloud, it will also be made available for on-premise, but the question is, do we really encourage the customer to use bgPF in on-premise.

T: Ok, I think I asked all my questions. Thank you for taking the time for the interview, it will really help me to write my project paper. I will now create a transcript and send it over to you for review, that I didn't missunderstand anything.

R: Ok, you are very welcome. I wish you have a gread project!

A.3. Internes Wiki bgPF

How to use bgPF?

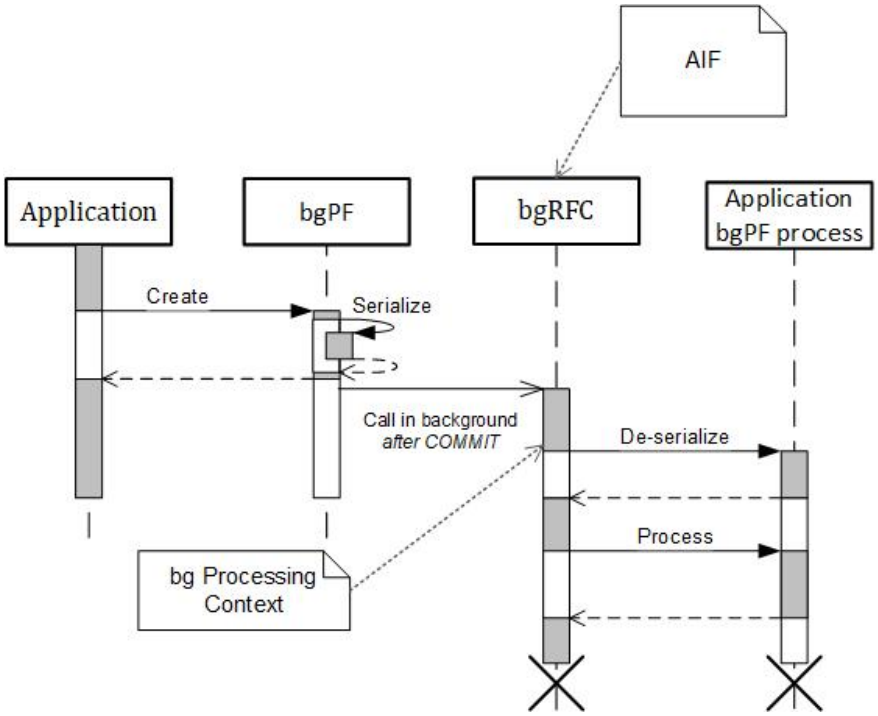
1. What is the background Processing Framework (bgPF)?

You probably know bgRFC. If we ignore the "R" (remote), then the bg(R)FC allows to trigger asynchronous processing in the background. This decoupling improves performance of the leading SAP LUW and makes it more robust. The bgPF is the modern ABAP Cloud variant of the bgRFC which combines many qualities:

- **Consistency**
  Transactional consistency and robustness are essential to ensure data consistency. The bgPF helps to outsource critical coding from actual SAP LUW. Within bgPF transactional consistency can be ensured if needed.
  bgPF supports transactional background processing with service quality "exactly once". An application can provide its own execution context object, to allow e.g. customers to define application specific background process priorities. A default execution context object is part of the bgPF.
  bgPF supports also queued background processing with service quality "exactly once in order". For this the application must provide its own execution context object. This is needed because there is no mechanism to guarantee uniqueness of queues.
- **End-user Performance**
  Better response time for actual SAPLUW, if post-processing can run in a different session asynchronously.
- **Scalability**
  For scalability of an application, it is better to split big processing steps into smaller chunks of work which can be distributed across work processes. The application server will use CL_SSI_DISPATCH in later releases to scale, this is done with the new bgRFC scheduler. Only if application coding is extracted and executed asynchronously in smaller chunks, then mechanisms like auto-scaling can be applied.
- **Tooling integration**
  ADT Cross Trace, AIF, bgRFC monitor can help you during development or later for monitoring your application.

Frameworks like e.g. OData and Events use bgPF for asynchronous and reliable execution.

1.1. How does it works?



- The first step for an application is to implemented one of our interfaces in an own class. At runtime the application calls the bgPF with an instance this class.
- Internally the bgPF serialize the class and call the bgRFC.
- The next COMMIT will trigger the bgRFC and a new ABAP session will start in the application server.
- bgPF will deserialize your class with all the data and will call the execute method of you implementation.

The AIF integration is optional.

2. When to use bgPF and when to use local business events?

Business events are modeled entities with a clear semantic (e.g. SalesOrderCreated, SalesOrderAccepted). We distinguish between provider and consumer. Typically these are different parties.

bgPF is used to execute business logic in the background. This has no modeling aspects and the data which is required is very specific for the concrete use case.

Both benefits from decoupling the background processing from the leading LUW, by improving the leading LUW's performance and robustness.

3. How to create your own implementation?

You have to implement interface IF_BGMC_OP_SINGLE or IF_BGMC_OP_SINGLE_TX_UNCONTR. The implementation of the AIF interfaces are optional.

Your operation implementation needs some data to do the processing in the background. It is up to you how you transfer the data to your object. E.g. our sample implementation (CL_BGMC_TEA_OPERATION) uses the constructor.

```
->set_operation( NEW CL_BGMC_TEA_OPERATION_TX_CON( exporting es_data = value #( ... )  )
```

3.1. Interfaces

1. IF_BGMC_OP_SINGLE
2. IF_BGMC_OP_SINGLE_TX_UNCONTR
3. IF_BGMC_OPERATION_AIF & IF_BGMC_OPERATION_AIF_CONF

3.1.1. IF_BGMC_OP_SINGLE (default) - Transactional control

The transaction controlled interface should be implemented per **default**. ABAP will make sure, that the basic rules of the SAP LUW are followed. We introduced the first checks in RAP and extracted the minimal restrictions which are valid in all ABAP transactions into something RAP-independent which is now also natively integrated into bgPF:

- Modify phase
  - No DB modifications on the primary connection are allowed.
  - (Implicit) DB-COMMIT is allowed. (Be aware that http communication in ABAP does an implicit DB-COMMIT)
  - no update task function module allowed
- Save phase (started via call of CL_ABAP_TX=>SAVE() in your implementation)
  - DB modifications are allowed.
  - update task function modules are allowed
  - No (implicit) DB-COMMIT is allowed.

Violations of the transactional contract and will lead to dump (or be logged). The advantage of the transaction controlled interface is that implicit database commits that can are apply by calling other parts are forbidden and controlled by the ABAP runtime. ACID rules that must be fulfilled for the SAP LUW are guaranteed.

Hint: The checks are done if checkpoint group CC_STMT is set to "Abort" (tx SAAB). This check is enabled in Steampunk and S/4 development systems. More details here: Checkpoint Groups - Steampunk - Wiki@SAP

3.1.2. IF_BGMC_OP_SINGLE_TX_UNCONTR - Without transactional control

The transaction uncontrolled interface should be implemented, when you can't follow the rules of the SAP LUW. This is only applicable in specific scenarios - e.g. using the XCO library to activate several DDIC / CDS artifacts where several COMMITs occur (which cannot be changed). An operation, that is not transactional controlled can be implemented freely. E.g. you can do COMMIT WORK or ROLLBACK WORK inside your implementation.

3.1.3. IF_BGMC_OPERATION_AIF* - AIF interfaces (Optional)

For an AIF integration you need to implement two interfaces:

IF_BGMC_OPERATION_AIF - Is used to transfer data to AIF and vice versa. GET_INPUT() and SET_INPUT() needs to be implemented.
IF_BGMC_OPERATION_AIF_CONF - This configuration (design time) interface for operations must be implemented to create a data container for AIF. GET_INPUT_CONTAINER() needs to be implemented.

3.2. Sample implementation

- CL_BGMC_TEA_OPERATION_TX_CON - Implementation of the transaction controlled interface
- CL_BGMC_TEA_OPERATION_TX_UNCON - Implementation of the transaction uncontrolled interface
- CL_BGMC_TEA_OPERATION - Implementation of AIF interfaces.
- CL_BGMC_TEA_APPLICATION is a sample application that use the bgPF to execute the implementation CL_BGMC_TEA_OPERATION*.

3.3. Can I do an automatic retry if an error happen?

In specific error situation (like e.g. when a database LOCK cannot be set) it is possible for an operation to trigger a future retry of the background process when it raises an exception. In your implementation you have to set the attribute TYS_RETRY_SETTINGS of the exception CX_BGMC_OPERATION. The bgPF with your operation will be scheduled for a retry.

- A retry can be attempted up to 3 times.
- It currently only works for queued background processes.

Automatic Retry

```
" In your implementation of IF_BGMC_OP_SINGLE_TX_*

method if_bgmc_op_single_tx_contr~execute.
..
    raise exception new cx_my_bgPF_exception( textid         = cx_my_bgPF_exception=>t100_lock_problem
                                              retry_settings = value #( do_retry = abap_true ) ).
endmethod.
```

3.4. How to call a bgPF

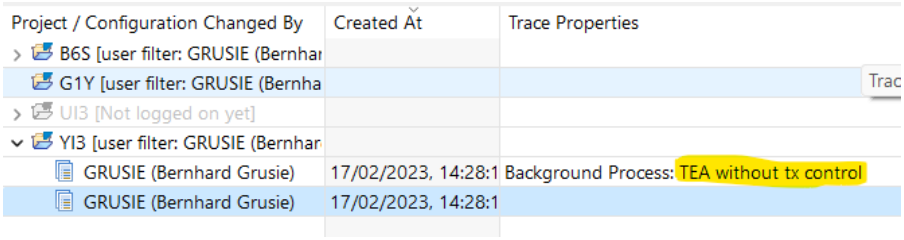
```
cl_bgmc_process_factory=>get_default(
                          )->create(
                          )->set_name( '<YOUR_TEXT>'
                          )->set_operation(    NEW <YOUR_IMPLEMENTATION_OF_IF_BGMC_OP_SINGLE>(   exporting
es_data = value #( ... ) )
                        //  )->set_operation_tx_uncontrolled( NEW
<YOUR_IMPLEMENTATION_OF_IF_BGMC_OP_SINGLE_TX_UNCONTR>( exporting es_data = value #( ... ) )
                          )->save_for_execution( ).

 COMMIT WORK. " If you use bgPF inside of RAP a commit will be done from RAP runtime. Without a commit, your
background operation will not start
```

An application instantiates synchronously a bgPF. Then the operation is added to the bgPF process. A save_for_execution() saves the process for later execution. Later on an explicit or implicit COMMIT triggers the execution of the new application session.

The default background processing context (get_default()) is shared with other applications and cannot be configured for specific applications. Or you use y our own bg Processing Context with: cl_bgmc_process_factory=>get('<NAME>'). For queued execution see How to use a Queue? To create an own Processing Context, see Own Processing Context. See also the current limitation in: bgPC_ADT

Method set_name('<YOUR_TEXT>') is optional. In the picture below we used "TEA without tx control". The use-case is to add runtime information e.g. "SalesOrder-42" to distinguish between different instances of SalesOrders calls of the bgPF. There is no AIF integration of this information so far.

When you use bgPF in RAP the method SAVE_FOR_EXECUTION() in this interface can only be called in the late save, see bgPF in RAP



3.4.1. Process Monitor

Method IF_BGMC_PROCESSSAVE_FOR_EXECUTION returns a monitor instance for the current background process. For most use cases this monitor is not needed.

It is intended for applications that have their own "book-keeping" and monitoring. Such an application might do something like this:

- In the current session it starts a background process and store the corresponding monitor instance in its bookkeeping table.
  It can use method IF_BGMC_PROCESS_MONITORTO_STRING to serialize the instance.
- At a later point in time (in a new session) the application needs to check the state of the background process.
  It re-instantiates the monitor instance via CL_BGMC_PROCESS_FACTORYCREATE_MONITOR_FROM_STRING and uses it to check the state of the background process.

3.4.2. How to use a Queue?

See limitation in: bgPC_ADT

To use the service quality "Exactly Once and In Order" (EOIO) , you have to use the queue method in the factory. The queue name is freely choosable from you. Our recommendation would be to use the key of your object as queue name. To avoid a clash for the queue name with other applications, you must us an own background Processing Context. Therefore, the default background Processing Context can't be used with a queue.

Queue in bgPF

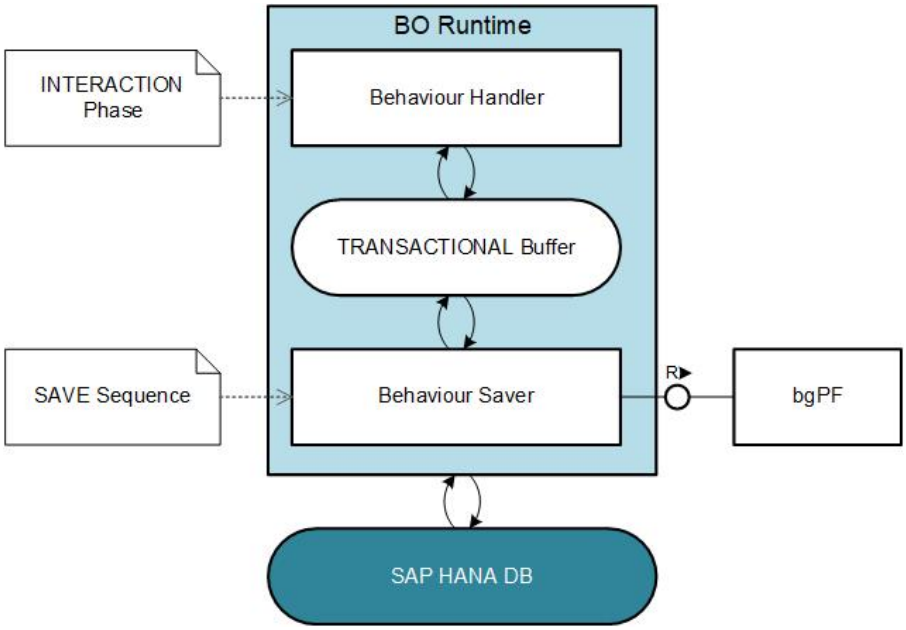
```
cl_bgmc_process_factory=>get_for_queued( iv_bg_processing_context = '<YOUR_bgProcessingContext'
                                         iv_queue                 = 'QUEUE_1' ).
```

3.4.3. How to use bgPF & RAP?

3.4.3.1. Use bgPF inside RAP

The RAP business object runtime mainly consists of two parts:

The first part is the **interaction phase**, in which a consumer calls the business object operations to change data and read instances with or without the transactional changes. The business object runtime keeps the changes in its internal transactional buffer which represents the state of the instance data. This transactional buffer is always required for a business object. After all changes were performed, the data can be persisted. This is realized with the **save sequence.**



The bgPF method save_for_execution( ) must be called in the save sequence. You can call all bgPF methods inside the save sequence. As alternative you can create the operation and set it into the bgPF process in the interaction phase. Method save_for_execution( ) must be called on the same bgPF instance as in the interaction phase.

SAP help: The RAP Transactional Model for the SAP LUW / behaviour handler / behaviour saver

bgPF call in RAP

```
" E.g. modify handler
CLASS <CL_MY_BEHAVIOUR_HANDLER> DEFINITION INHERITING FROM cl_abap_behavior_handler.
  METHODS my_modify_method FOR MODIFY
      IMPORTING it_param FOR ACTION <MY_BO>~my_action RESULT result.

" E.g. SAVE sequenz
CLASS <CL_MY_BEHAVIOUR_SAVER> DEFINITION INHERITING FROM cl_abap_behavior_saver.
  PROTECTED SECTION.
    METHODS save REDEFINITION.  "save_for_execution() must be called here
```

3.4.3.2. Use RAP inside bgPF

Your bgPF process is executed in a new ABAP session. The Entity Manipulation Language (EML) is a part of the ABAP language that enables access to RAP business objects. Here you can use the EML inside of bgPF to read or modify RAP Business Objects as usual. If you use the IF_BGMC_OP_SINGLE_TX_CONTR you have to follow the transaction buffer and commit rules that are explained here: IF_BGMC_OP_SINGLE_TX_CONTR

3.4.4. When do I need an own background Processing Context and how can I create such an object?

Important hint: The delivery of the background Processing Context will be done with 2402. At the moment the only option is to use the default destination from bgPF in the Cloud (CE-0326 - Background RFC (bgRFC) - Technical Configuration). The name of the bgRFC destination is BGPF.

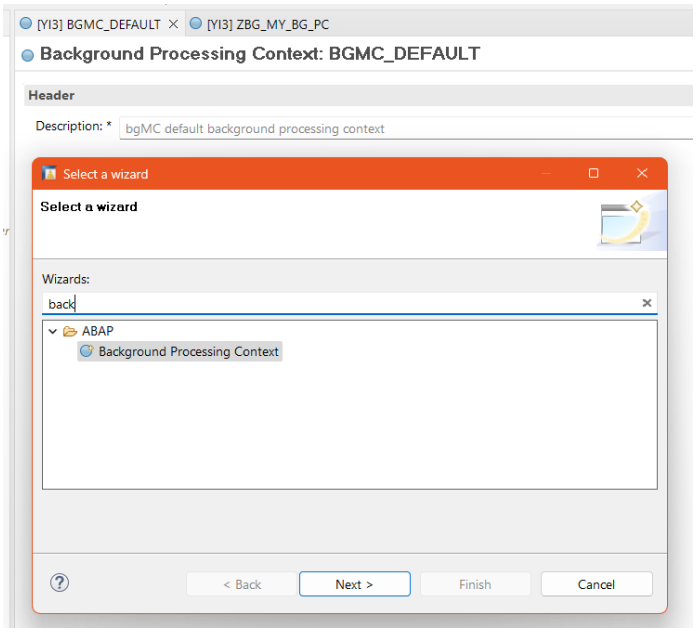
OnPremise customers have to create an bgRFC inbound destination with the name BGPF in their systems. To achieve this they have to configure a BGRFC inbound destination with transaction SBGRFCCONF.

Therefore, "In Order" processing is not possible with 2308 & 2311

If you application needs "In Order" processing you need an own background Processing Context object, see How to use a Queue?.

The idea behind an own object is that in the future (not with 2308) customers can define priorities with this object. For sure you can use the default Processing Context that is delivered with bgPF. All applications that use this default object have the same priority and customers can't be distinguishing between different applications. This is relevant if mass data are processed in the system.

With ADT 3.33.300 you can create an own background Processing Context.



3.4.5. I am wondering why my background operation does not start. What could be the problem?

1. You can use the trace to see if the background operation from you was started, see ABAP Cross Trace
2. If not, ensure that a COMMIT WORK was called. The RAP runtime will do a commit.

3.5. How to write Unit tests?

You can use class CL_BGMC_TEST_ENVIRONMENT to create a test environment. A spy is created by CL_BGMC_TEST_ENVIRONMENT=>CREATE_FOR_SPYING( ).

- An application's unit test can inspect the processes its productive coding generates.
- The tests validate, that the application's operation is instantiated and added to a process correctly.
- The operation themselves is tested independently.

Spy in action

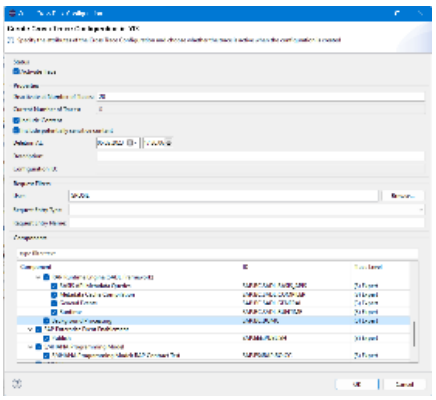
```
data(lo_bg_spy) = CL_BGMC_TEST_ENVIRONMENT=>CREATE_FOR_SPYING( ).

mo_cut->if_bgmc_tea_application~run_single_tx_controlled( ms_operation_input ).

lo_bg_spy->assert_number_of_processes( 1
            )->get_process( 1
            )->assert_is_saved_for_processing(
            )->assert_number_of_operations( 1 ).
```

4. Monitoring

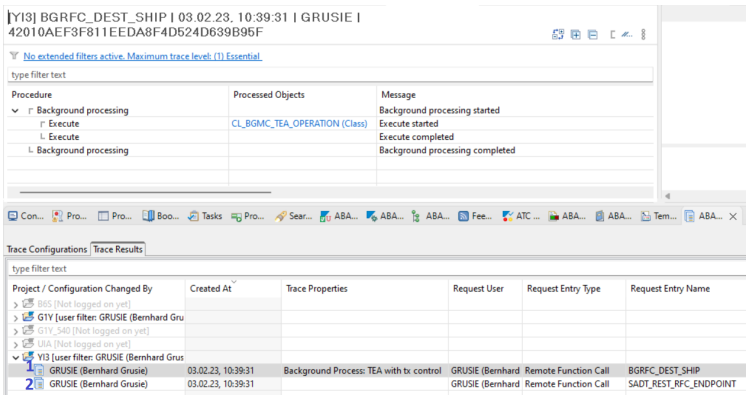
4.1. ADT ABAP Cross trace

background Process Framework in ABAP Cross Trace.

1. Create Configuration:

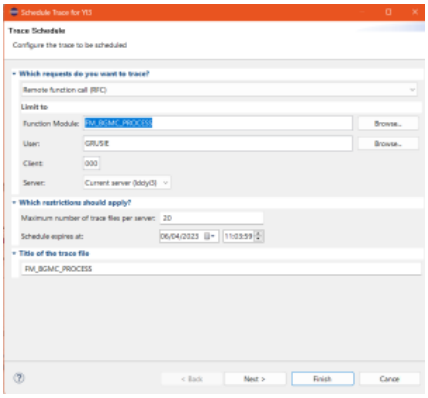


2. Trace Result:
Entry two is from the synchronous call. Entry one is from the new background process

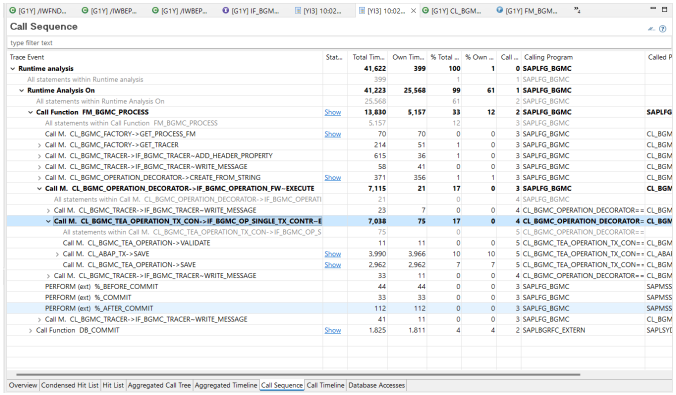


4.2. ADT Performance Trace

If you like to find the bgPF in the ADT Performance trace you can use Function Module FM_BGMC_PROCESS as filter:



In the result list you will have at least to entries for the background process and for the caller of the bgPF. In the call sequence you will find you operation implementation, see the marked entry in the picture below:



4.3. bgRFC monitor

With transaction SBGRFCMON you can start the bgRFC monitor. As alternative you can use the Fiori App: SAP Help

4.4. AIF integration

The AIF integration needs beside the implementation of the bgPF AIf interfaces, also AIF customizing. You have to follow the the AIF how to guide: AIF Interface Creation for Monitoring bgPF

4.5. How to deal with dumps?

If you operation implementation leads to a dump the monitoring tools like ADT Cross Trace or AIF (

FBSAIF-2093 - Getting issue details... STATUS

) are not aware of this dump. With 2308 there is no solution from bgPF side for this problem.

When you have an own monitoring tool, you can use the EPP Passport to find in the dumps your operation.