Prozess Mining in S/4 HANA

Echtzeit-Visualisierung eines Einkaufsprozesses

Tizian Cockx
DXC Technology
Hewlett-Packard-Straße 1
61352 Bad Homburg
tcockx@dxc.com

Norbert Ketterer
Hochschule Fulda
Leipziger Str. 123
36037 Fulda
Norbert.Ketterer@cs.hs-fulda.de

ABSTRACT

In dieser Arbeit wird untersucht, in wieweit sich ein graphisches, interaktives Prozess Mining Tool mit Technologien, die über die Entwicklungsplattform im Standard einer aktuellen SAP S/4 HANA Version mit ausgeliefert werden, erstellbar ist. Es werden insbesondere die Fragen einer effizienten Datenextraktion der Prozessaktivitäten sowie einer benutzerfreundlichen, graphischen Darstellung der Ergebnisse des Mining-Laufs sowie der Implementierungsaufwand, der sich im Zielsystem ergibt, betrachtet. Bezüglich der Mining-Grundlagen orientiert sich diese Arbeit stark an Van der Alst [1]. Bei den betrachteten Prozessen handelt es sich im Prototypen um zwei Prozesse des Einkaufs. Die Arbeit basiert auf eine Abschlussarbeit, die bei der DXC technology GmbH unter der Betreuung der Hochschule Fulda erstellt wurde.

Keywords

Prozess Mining, SAP UI5, S/4 HANA, Core Data Services

EINLEITUNG

Geschäftsprozesse, laufen Sie nun in digitaler Form in betrieblichen Anwendungen ab oder auch rein analog in Papierform, bilden die Basis für jede unternehmerische Tätigkeit. Aufgrund dieser zentralen Rolle sollte grundsätzlich darauf geachtet werden, dass die Geschäftsprozesse in der gewünschten Form ablaufen und mit den gegebenen Prozesszielen konform sind. Der Ablauf der Geschäftsprozesse kann heutzutage in einem Anwendungssystem mit Prozess Mining Verfahren analysiert werden, deren Ziel es ist, aus den Aufzeichnungen der Vorgänge im System (Event-Logs) Erkenntnisse zu gewinnen, um darauf basierend Aktionen abzuleiten [1]. ERP-Systeme, als integrierte unternehmensweite Anwendungssysteme, die zur Koordination der wesentlichen internen Prozesse eines Unternehmens dienen, bilden eine relevante Datenquelle für Prozess Mining, da hier ein hoher Prozentsatz der Geschäftsprozesse abgewickelt wird und somit in den Datenbanktabellen des ERP-Systems die Prozesse relevante Daten (oftmals in diesem Zusammenhang auch als auch als „Spuren“ bezeichnet - z.B. wiederum in [1], hier Kapitel 2.2) für die Analyse hinterlassen. S/4 HANA ist ein ERP-System, welches auf aktuellen Technologien basiert (In-Memory Datenbank, HTML-5/ Javascript Technologien als SAP UI5 basierter Frontend), die einerseits Performancevorteile bei der Extraktion der Prozessdaten erwarten lassen und andererseits für den Benutzer aufgrund der technologischen Plattform einen webbasierten interaktiven Miningablauf in Aussicht stellen; dieses ist aber kundenspezifisch zu implementieren.

Der Fokus der Arbeit liegt nicht auf dem Prozess Mining selbst, sondern in der Evaluierung eines Implementierungsansatzes in S/4 HANA. Auch werden verwendete Basistechnologien, wie z.B. SAP Gateway Services [2], kurz anhand ihres Funktionsumfangs betrachtet. Es sollen hier nicht alle Facetten des Prozess Minings betrachtet werden, so wie sie in [1] beschrieben sind, sondern es soll ein spezifischer Algorithmus aus dieser Quelle implementiert werden. Zur Abgrenzung des Prozessrahmens beschränkt sich die Extraktion auf zwei spezifische Einkaufsprozesse; das für diese Prozesse Gesagte kann allerdings dann analog auf andere Prozesse verallgemeinert werden - es verändert sich lediglich die Datenextraktion. Das Untersuchungsziel der Arbeit stellt die Realisierbarkeit der Event-Log Extraktion von der Datenbank des S/4 HANA Systems dar, ergänzend werden prototypisch Lösungsansätze entwickelt, um das erstellte Prozessmodell mit Prozess Mining Verfahren zu erstellen und anschließend mit Hilfe von Bausteinen der UI5-Bibliothek des S/4 HANA Systems zu visualisieren. Bezüglich der Extraktion soll auch die Performance prinzipiell betrachtet werden; unter Berücksichtigung der Tatsache, dass das Coding nicht in einer Produktivumgebung mit entsprechenden Massendaten getestet werden kann.

PROBLEMDEFINITION

Extraktion der Datenbasis für das Mining

Nach Österle ([3], S. 10) ist ein Geschäftsprozess eine Abfolge von Aufgaben, die über mehrere organisatorische Einheiten verteilt sein können und deren Ausführung von informationstechnologischen Anwendungen unterstützt wird. In betrieblichen Anwendungen führen diese Aktivitäten zu Ereignissen (etwa der Anlage einer Bestellung), die prinzipiell extrahierbar sind, sei es durch Extraktion expliziter Änderungsinformation oder durch Abgleich von Datenbankabzügen (etwa der „Differential Snapshot Problematik“) - sehr Vergleichbar der Datenextraktion aus dem Bereich des „Da-

ta Warehousing“. Ein in diesem Zusammenhang extrahiertes Event-Log besteht üblicherweise aus einer Sammlung von Ereignisdaten, die bei der Ausführung einer Aktivität im Prozess aufgezeichnet werden [1], Kapitel 2.2 und 5.2. Diese Ereignisdaten sind dabei so zu speichern, dass ein Mining der Prozesse stattfinden kann. Die Extraktion sollte möglichst so erfolgen, dass sie Online als Teil des Mining-Prozesses erfolgen kann.

Analyse der Prozessdaten

Da ein Geschäftsprozess eine Folge von Aktivitäten ist, die unternehmensrelevante Ereignisse erzeugen, sind genau diese Aktivitäten zu extrahieren. Da aber von jedem Geschäftsprozess eine Reihe von individuellen Instanzen durchlaufen werden, sind die Aktivitäten diesen Instanzen zuzuweisen; es findet somit eine Gruppierung nach diesen Instanzen statt [1], Kapitel 5.2. Die Aufgabe von Prozess Mining ist es, Wissen aus Event-Logs heutiger Systeme zu extrahieren - dieses Wissen kann dann auf verschiedenste Arten verwendet werden - beispielsweise um die Basis für Ansätze von Prozessverbesserungen zu liefern. Das wesentlich zu lösende Problem beim Prozess Mining besteht dann darin, auf Basis der Event-Logs verschiedene Prozessinstanzen zu identifizieren und miteinander zu vergleichen - dieser Vergleich liefert dann Aufschlüsse über den konkreten und in den verschiedenen Instanzen potentiell unterschiedlichen Ablauf der Aktivitäten. Die Abfolge der Aktivitäten, die sich auf eine bestimmte Prozessinstanz bezieht, wird als **„Trace“** bezeichnet [1], Kapitel 3. Ein Beispiel von zwei unterschiedlichen Traces, die aber beide auf dem gleichen Prozess basieren, zeigen die Abbildungen 1 und 2. Es sind auf Basis der extrahierten Event-Logs diese Traces zu finden und gegenüber zustellen.

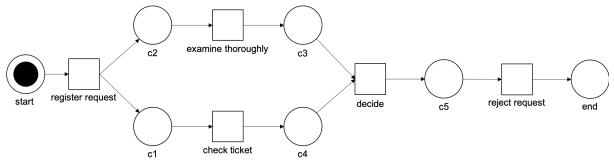

Abbildung 1: Bearbeitung Entschädigungsantrag bei Fluggesellschaft in Anlehnung an [1], Kapitel 3

Die durch das „AND“ eingeleitete Parallelität der beiden Vorgänge „examine thoroughly“ und „check ticket“ ist nicht synchronisiert und kann dann im Ablauf der Prozessinstanz zu verschiedenen Reihenfolgen der Ausführung dieser beiden Vorgänge führen, wie in Abbildung 2 zu erkennen ist.

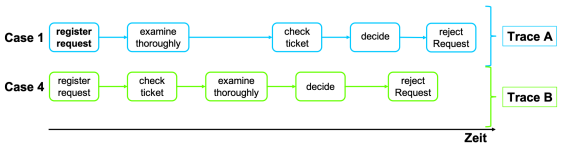

Abbildung 2: Instanzen von Traces für Entschädigungsantrag

STAND DER TECHNIK

Extraktion der Datenbasis für das Mining

Grundlagen der Datenextraktion

Um den Ablauf eines Traces in eine geordnete Reihenfolge zu bringen, wird in einem Event-Log zusätzlich ein Zeitstempel (Timestamp) benötigt ([1], Kapitel 5.2) - etwa wie in Tabelle 1 gezeigt. Am Ende wird jeder Eintrag in einem Event-Log durch eine Prozessinstanz, eine Aktivität und einen Zeitstempel beschrieben, zusätzliche Information mit Bezug zu den Ereignissen sind ebenfalls denkbar. Im Falle eines Einkaufsprozesses kann eine Instanz dann durch die entsprechende Belegnummer des Einkaufsobjekts (Bestellanforderung, Bestellung, etc.) repräsentiert werden, es ist dabei darauf zu achten, dass ein Beleg bei einem vollständigen Prozessdurchlauf auch Folgebelege besitzt, etwa im Falle des Einkaufs den Beleg zur Bestellanforderung, dann Bestellung dann Avise, etc.

Table 1: Event-Log Beispiel mit zwei Fällen [4]

Prozessinstanz	Aktivität	Zeitstempel	Mitarbeiter
1	register request	30-12-2010:11.02	Pete
1	examine thoroughly	31-12-2010:10.06	Sue
1	check ticket	05-01-2011:15.12	Mike
1	decide	06-01-2011:11.18	Sara
1	reject request	07-01-2011:14.24	Pete
4	register request	06-01-2011:15.02	Pete
4	check ticket	07-01-2011:12.06	Mike
4	examine thoroughly	08-01-2011:14.43	Sean
4	decide	09-01-2011:12.02	Sara
4	reject request	12-01-2011:15.44	Ellen

Datenextraktion in S/4 HANA

Die Datenextraktion befasst sich mit der Bereitstellung eines Event-Logs, welches zur späteren Weiterverarbeitung mit Hilfe von Prozess Mining Algorithmen benötigt wird. Mit Auslieferung von S/4 HANA steht in der entsprechenden Entwicklungsplattform dieses ERP-Systems ein neues virtuelles Datenmodell (VDM) bereit, welches als architekturelle Grundlage für die meisten Anwendungen innerhalb von S/4 HANA dient. Beispielhaft ist das neue ABAP Programmiermodell für SAP Fiori [5] zu nennen, bei dem das VDM für das Datenmodell und die Datenbereitstellung zuständig ist. Eine Umsetzung des VDM erfolgt mit Hilfe von SAP Core Data Services (SAP CDS) bei denen die Datenmodellierung mit Hilfe der CDS Data Definition Language (CDS DDL) umgesetzt wird. Die CDS DDL lehnt sich stark an die SQL an und erlaubt Entwicklern die CDS Datenmodellierung. Unter dem VDM ist jedoch keine unstrukturierte Ansammlung von CDS Views zu verstehen, sondern es steht eine klare Struktur dahinter. Nachfolgend sollen die innere Struktur und die Schichten des VDM näher erläutert werden. Ein Gesamtüberblick der Architektur des VDM ist in der Abbildung 3 zu finden.

Interface Views

Basierend auf SAP HANA Datenbanktabellen bilden sogenannte „Interface Views“ die unterste Schicht innerhalb des Datenmodells. Als wichtigster Bestandteil des VDMs sollen sie zukünftig nicht von Updates bzw. Patches betroffen sein. Ausschließlich Views aus dieser Schicht greifen auf die zugrundeliegenden Datenbanktabellen zu und repräsentieren zumeist einzelne Entitäten, wie z.B. Warengruppen

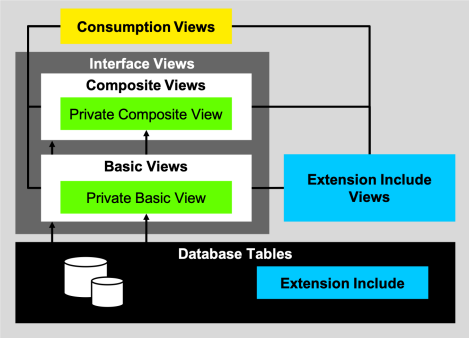

Abbildung 3: SAP CDS Architektur [6]

oder Lieferanten. CDS Annotationen reichern die SQL Logik der CDS Views mit zusätzlichen Metadaten an [7], eine Interface View wird immer mit der CDS Annotation @VDM.viewType: #BASIC versehen. Bei der Datenselektion findet zusätzlich eine Semantik Anreicherung statt, um die betriebswirtschaftliche Sicht nachvollziehen zu können. Umgesetzt wird diese Anreicherung, indem Datenbankfeldern mit Hilfe von Aliassen sprechende Identifikatoren zugeordnet werden. Eine Beispielimplementierung einer CDS View ist anhand des CDS „I_MaterialGroup“ in Listing 1 zu finden. Interface Views können jedoch auch kundenspezifisch erstellt werden.

Listing 1: Beispiel CDS View „I_MaterialGroup“, zur Selektion von Warengruppen.

```
@AbapCatalog.sqlViewName: 'IMATGROUP'
@Analytics: { dataCategory: #DIMENSION, dataExtraction.enabled: true }
@VDM.viewType: #BASIC
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Material Group'
@Search.searchable: true
@ObjectModel.representativeKey: 'MaterialGroup'
@ObjectModel.usageType.serviceQuality: #A
@ObjectModel.usageType.sizeCategory : #S
@ObjectModel.usageType.dataClass: #MASTER
@ClientHandling.algorithm: #SESSION_VARIABLE

define view I_MaterialGroup
  as select from t023
  association [0..*] to I_MaterialGroupText as _Text on $projection.
        MaterialGroup = _Text.MaterialGroup
{
    @ObjectModel.text.association: '_Text'

    @Search.defaultSearchElement: true
    @Search.fuzzinessThreshold: 0.8
    key cast(t023.matkl as productgroup) as MaterialGroup,
    /* Associations */
    _Text
}
```

Consumption Views

Eine Consumption View CDS (oftmals auch nur als „Consumption View“ bezeichnet) ermöglicht den Zugriff auf die Daten eines Business-Objekts bzw. eine Menge von Tabellen - etwa durch einen ODATA-Service (zur Veröffentlichung von CDS-Information über ODATA siehe z.B. [8]). Sie richten sich somit an individuelle Konsumenten der Daten des VDM und tragen die CDS-Annotation @VDM.viewType: #Consumption. Die oberste Schicht der Views setzt sich wiederum aus mind. 1 Interface Views zusammen; es gilt die Leitregel, das Datenbanktabellen nicht direkt von Consumption Views angesprochen werden dürfen [6]. Weiterhin ist zu beachten, dass sie nicht für die Wiederverwendung in anderen Consumption Views bestimmt sind, eine Ausnahme bilden analytische Queries, bei der die Mehrfachverwendung möglich ist [7].

	a_1	a_2	a_3	...
a_1				
a_2				
a_3				
...				

Table 2: Übergangsmatrix des Kontrollflussalgorithmus nach [9]

Extension Views

„Extension Views“ können sowohl für „Interface Views“ als auch „Consumption Views“ angelegt werden. Sie ermöglichen es Kunden und Entwicklern bestehende CDS Views zu erweitern. Bei der CDS View Erweiterung stehen zwei unterschiedliche Erweiterungsoptionen zur Auswahl: CDS View Erweiterungen und CDS Metadaten Erweiterungen [7]. Dabei dienen CDS Metadaten Erweiterungen der Anreicherung von aktiven Annotation eines CDS Datenmodells, oder zur Übersteuerung bereits vorhandener Annotationen von CDS Views. Ziel der CDS View Erweiterung ist es bestehende Datenmodelle um zusätzliche Felder bzw. Assoziationen zu erweitern, um diese den Anforderungen des Kunden anzupassen.

Prozess Mining Verfahren

Der wesentliche Aspekt in dieser Arbeit zum Prozess Mining wird auf die Ableitung des **Kontrollflusses** gelegt. Ein einfaches Verfahren hierzu stellt der „Kontrollflussalgorithmus“ zur Erzeugung der „Übergangsmatrix“ dar (siehe hierzu [9], Kapitel 2). Der Kontrollflussalgorithmus beschreibt dabei ausschließlich ein sequentielles Abarbeiten eines Event-Logs - er erzeugt im Rahmen seines Ablaufs dann die „Übergangsmatrix“. Es gilt besonders darauf zu achten, dass das Event-Log eine aufsteigende Sortierreihenfolge nach Prozessinstanz, gefolgt vom Zeitstempel, befolgt. Vor der Analyse eines Event-Logs ist nicht bekannt, wie viele Übergänge tatsächlich stattgefunden haben, so dass davon ausgegangen werden muss, dass jeder Übergang zwischen diesen Aktivitäten möglich ist. Wenn in dem Event-Log nun N Aktivitäten existieren, dann können somit N^2 Übergänge zwischen diesen Aktivitäten vorkommen.

Basis dieses Algorithmus bildet die Übergangsmatrix aus Tabelle 2, die die Anzahl der Übergänge zweier Aktivitäten listet. Die Matrix wird erzeugt, indem in Zeile und Spalte sämtliche Aktivitäten des Logs eingetragen werden. Die Zellen der Matrix beinhalten dann die Gesamtzahl der Übergänge der Aktivität der Zeile zur Aktivität der Spalte, die im Event-Log identifiziert werden konnte. Die Matrix enthält somit die Anzahl der Übergänge zwischen allen denkbaren Kombinationen von Aktivitäten im Event-Log.

Der Kontrollflussalgorithmus zum Aufbau der Übergangsmatrix lässt sich dann einfach wie folgt beschreiben (in dem Listing wird lediglich die Matrix selbst generiert, welche dann in die Darstellungsebene übergeben werden kann):

Listing 2: Kontrollfluss-Algorithmus in Anlehnung an [9].

```
1  Let M be a square matrix of size |T|^2
2  Initialize M_{i,j} ← 0 for every position (i,j)
3  for each case id in the event log do
```

```
4       for each consecutive task transition a_i → a_j in
            that case id do
5       M_{i,j} ← M_{i,j} + 1
6       end for
7   end for
```

Prinzipiell müssen die folgenden beiden Punkte bei Generierung der Übergangsmatrix beachtet werden:

- Die Übergangsmatrix hängt stark von der Reihenfolge der Prozessinstanzen im Event-Log ab - insbesondere der Zeitstempel ist hier zu beachten, auch da der Zeitstempel nicht immer aus den Business-Objekten selbst vollständig ablesbar ist
- Die Form der Generierung nach diesem einfachen Algorithmus kann zu Rauscheffekten führen, da es viele mögliche Prozesspfade gibt. Aus diesem Grund werden in Mininganwendungen manchmal auch komplexere Algorithmen eingesetzt, wie etwa der $\alpha-Algorithmus$ - tendenziell würden dann semantisch hochwertigere Prozessmodelle erzeugt werden, die nicht nur für Knoten die möglichen Nachfolger gemäß Event-Log anzeigt, sondern echte logische Verknüpfungen in den Prozessmodellen. Eine Alternative zur Erzeugung semantisch hochwertigerer Prozessmodelle wäre dann eine Filterung auf verschiedene Varianten von Prozessdurchläufen.

Abhängigkeit vom Zeitstempel

Der Zeitstempel liefert für die Aktivitäten implizit eine Reihenfolge und spielt deshalb eine zentrale Rolle bei der Interpretation des Event-Logs. Tabelle 3 zeigt ein typisches Beispiel eines Extraktes eines Event-Logs nach [4] in dem die Abhängigkeit vom Zeitstempel gezeigt werden kann.

Table 3: Event-Log Beispiel mit zwei Fällen nach [4]

Instanz	Abstr. Aktivität	Aktivität	Zeitstempel
1	a	register request	30-12-2010:11.02
1	b	examine thoroughly	31-12-2010:10.06
1	c	check ticket	05-01-2011:15.12
1	d	decide	06-01-2011:11.18
1	e	reject request	07-01-2011:14.24
4	a	register request	06-01-2011:15.02
4	c	check ticket	07-01-2011:12.06
4	b	examine thoroughly	08-01-2011:14.43
4	d	decide	09-01-2011:12.02
4	e	reject request	12-01-2011:15.44

Table 4: Matrix für Tabelle 3

	a	b	c	d	e
a		1	1		
b			1	1	
c		1		1	
d					2
e					

Wird dagegen der Zeitstempel vertauscht, verändert sich die Matrix entsprechend, was dann zu einer anderen Prozessdarstellung führt.

Table 5: Event-Log Beispiel mit zwei Fällen nach [4] - umsortiert

Instanz	Abstr. Aktivität	Aktivität	Zeitstempel
1	a	register request	30-12-2010:11.02
1	b	examine thoroughly	31-12-2010:10.06
1	c	check ticket	05-01-2011:15.12
1	d	decide	06-01-2011:11.18
1	e	reject request	07-01-2011:14.24
4	c	check ticket	07-01-2011:12.06
4	a	register request	06-01-2011:15.02
4	b	examine thoroughly	08-01-2011:14.43
4	d	decide	09-01-2011:12.02
4	e	reject request	12-01-2011:15.44

Es ergibt sich dann die eine veränderte Übergangsmatrix, die in Tabelle 6 skizziert ist.

Table 6: Übergangsmatrix bei nicht korrekt eingehaltener Sortierreihenfolge.

	a	b	c	d	e
a		2			
b			1	1	
c	1	1		1	
d					2
e					

Verwendet man die Übergangsmatrixen der Tabellen 4 und 6 als Input für eine graphische Auswertung (hier dargestellt mit Hilfe des Tools „Disco“) ergeben sich selbstverständlich Unterschiede in den dargestellten Prozesspfaden, wie in Abbildung 4 und 5 zu erkennen ist.

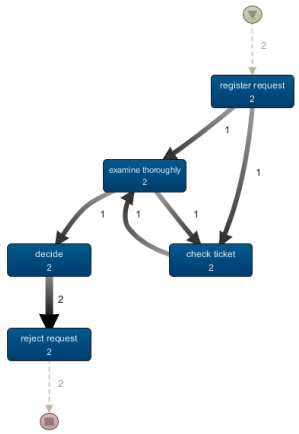

Abbildung 4: Ausgabegraph des korrekt sortierten Event-Logs

Visualisierung der Übergangsmatrix

Wird die Matrix mit einem Tool visualisiert, ist für jede Aktivität ein Knoten zu erzeugen und für jeden Zelleneintrag eine Kante zwischen Zeilen- und Spaltenaktivität - potentiell in einer Dicke, die die Anzahl der Übergänge repräsentiert - für Matrix 4 und Matrix 6 wären somit jeweils genau sieben Kanten vorzusehen, jedoch zwischen anderen Knoten und potentiell (je nach Visualisierungsplattform) in einer anderen Dicke; beispielsweise existiert eine Kante von Knoten

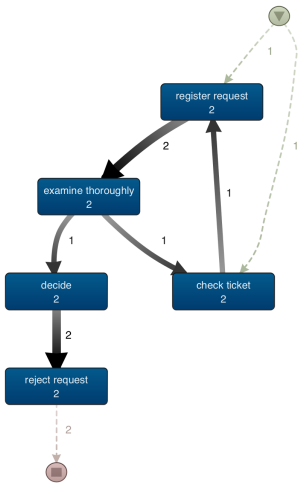

Abbildung 5: Ausgabegraph des nicht korrekt sortierten Event-Logs

a nach **b** dann potentiell in verschiedenen Dicken, um die Anzahl **Eins** bzw. **Zwei** zu repräsentieren.

α – Algorithmus
Der Kontrollflussalgorithmus zur Erstellung der Übergangsmatrix zählt lediglich die Übergänge zwischen Aktivitäten und konnte in dieser Arbeit verwendet werden, da die hiermit erstellte Matrix direkt an die graphische Ausgabe weitergegeben werden sollte - das Erkennen von Beziehungen erfolgt somit rein visuell. Ein einfacher Algorithmus, der bereits selbst Beziehungen zwischen den Aktivitäten findet, die über ein reines Abzählen hinausgehen, stellt der *α – Algorithmus* dar (siehe [1], Kapitel 6.2.1 und 6.2.2).

Ziel dieses Algorithmus ist das Extrahieren elementarer Beziehungen zwischen Aktivitäten (hier genannt **A** und **B**), die wie folgt definiert sind:

1. Eine direkte Vorgängerrelation R_1, die zwei Einträge **A** und **B** identifiziert, die im Log genau hintereinander stehen (dies entspricht dem Kontrollflussalgorithmus); also R_1**(A, B)**
2. Eine direkte Vorgängerrelation R_2, die eine Kausalität impliziert. Dies ist eine schärfere Forderung zu R_1, denn wenn gemäß R_1 gilt, daß R_1**(A, B)** und R_1**(B, A)**, kann dies gemäß dieser Relation nicht sein. Es gilt gemäß dieser Relation also nur **eine** Beziehung zwischen **A** und **B**. Hier spielt somit der Zeitstempel wieder eine entscheidende Rolle; hier wird somit eine echte Vorgänger-Nachfolger Beziehung ausgewiesen, die unabhängig von der Prozessinstanz ist.
3. Eine Relation R_3, die aussagt, daß die Elemente in keiner Beziehung zueinander gemäß R_1 stehen. Es gibt gemäß R_1 also weder eine Relation R_1**(A, B)** als auch R_1**(B, A)**. Diese Relation gilt jedoch auch immer zwischen der Aktivität mit sich selbst, also R_3**(A, A)**; dies drückt somit eine echte Exklusivität aus, wenn die Aktivität nicht identisch zu sich ist.
4. Eine Relation R_4, die aussagt, dass gemäß R_1 sowohl R_1**(A, B)** als auch R_1**(B, A)** gilt. Dies tritt beispielsweise immer dann auf, wenn aufgrund von Ungenauigkeiten des Zeitstempels die Reihenfolge der Aktivitäten **A** und **B** schwankt oder selbstverständlich auch, wenn im Ablauf eine zeitliche Reihenfolge nicht vorgegeben werden kann, beispielsweise, da beide Aktivitäten in Prozessmodell mit einem „AND“ verknüpft sind. Ein „AND“-Split innerhalb der Prozesslogik legt bekanntlich genau die Reihenfolge der Prozesspfade nach dem Split nicht fest; die Aktivitäten hinter dem Split in der verschiedenen Prozesspfaden können somit in beliebigen Reihenfolgen zueinander durchgeführt werden - etwa die „**B**“-Aktivitäten zu den „**C**“-Aktivitäten in Abbildung 6.

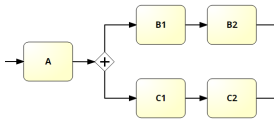

Abbildung 6: AND-Split mit zwei Prozesspfaden

Die Relation R_2 lässt sich dann direkt in eine Prozesssequenz umsetzen; Relation R_3 kann in ein „XOR“ umgesetzt werden, wenn die entsprechenden weiteren Relationen R_2 vorhanden sind; Relation R_4 endet mit entsprechenden Relationen R_2 in einem „AND“. Es werden somit die Prozesselemente wirklich mit Semantik belegt, nicht nur (wie in dieser Arbeit angestrebt) in eine Grafik umgesetzt, deren Elemente keine Semantik wie „AND“ oder „XOR“ tragen müssen. Da in einem ersten Schritt die Semantik nicht benötigt wird und auch bei einer Reihe von Prozessanalysetools diese Semantik nicht ausgewiesen wird, wurde dieser Algorithmus nicht implementiert. Ein Beispiel eines gefundenen „XOR“, welches auf zwei Beziehungen des Typs R_2 basiert (zwischen **A** und **B** sowie **A** und **C**) sowie einer Beziehung des Typs R_3 zwischen **B** und **C** zeigt Abbildung 7.

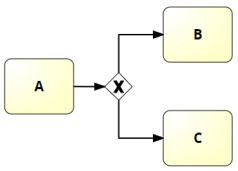

Abbildung 7: XOR basierend auf zwei bzw. einer gefundenen Relationen des Typs 2 bzw. 3

Prozess Mining Tools

Laut [1], Kapitel 6 werden bei Prozessmodellen zur Prozessentdeckung drei grundlegende Modelle unterschieden:

1. **Informelle Prozessmodelle** Prozessmodelle ohne formale Interpretation, die mit Traces aus einem Event-Log verknüpft werden können.
2. **Formale low-level Prozessmodelle** Dazu zählen zum Beispiel Übergangssysteme oder Markovketten.
3. **Formale high-level Prozessmodelle** Prozessmodelle, die Auswahlmöglichkeiten, Parallelität, Schleifen, etc. ermöglichen. Darunter zählen zum Beispiel BPMN Modelle, Petri-Netze oder Prozessbäume.

Ein Algorithmus zur Erzeugung einer Übergangsmatrix kann dann selbstverständlich nur die Basis für ein low-level Prozessmodell darstellen, während für ein high-level Prozessmodell der Algorithmus mehr Semantik aus dem Log extrahieren muss, wie dies in Abbildung 7 geschieht. Informelle und low-level Prozessmodelle bilden einen einfachen Einstieg, um Prozess Mining Techniken nachvollziehen zu können. Sie haben aber ein Problem, die Parallelität von Prozessen klar darzustellen, da ein paralleler Prozess lediglich in einer Reihe verschiedener Ausführungsreihenfolgen zwischen Aktivitäten resultiert, was letztlich in einer sehr umfangreichen Darstellung des Prozesses mündet; statt einfach im Prozessmodell die betroffenen Aktivitäten durch ein „AND“ zu verknüpfen und somit die Parallelität explizit zu modellieren werden sämtliche Durchläufe dargestellt, die in der verschiedenen Prozessinstanzen auftraten und somit im Event-Log festgehalten wurden.

Im folgenden sollen eine Reihe gängiger Tools bezüglich der folgenden Merkmale im Sinne des Fokus dieser Arbeit betrachtet werden:

1. Art des Modells: informell, low-level oder high-level
2. Echtzeiteinbindung in Anwendungssoftware
3. Einbindung in S/4 Hana mit graphischer Auswertung

Celonis

Das Mining-Tool Celonis ermittelt auf Basis der Anzahl der Durchläufe Prozessvarianten und Kennzahlen dazu - ein Beispiel einer Prozessvariante aus dem „Variant Explorer“ zeigt Abbildung 8. Wie man dort erkennen kann, werden die Häufigkeiten der Übergänge zwischen den Aktivitäten angegeben (ähnlich der Information in der Übergangsmatrix), es werden jedoch keine logischen Prozesselemente ermittelt, wie dies im Zusammenhang mit dem $\alpha - Algorithmus$ skizziert wird. Neben den Prozessvarianten werden auch Kennzahlen ermittelt, wie in Abbildung 9 dargestellt wird. Es existieren Interfaces, um Daten von SAP-Systemen nach Celonis zu extrahieren - beispielsweise scheint der Prozess in Abbildung 9 auf IDES-Daten zu basieren. Jedoch läuft das Tool nicht direkt in einer S/4 HANA Landschaft.

ProM

Ein Großteil der bisher entwickelten Prozess Mining Verfahren ist in ProM umgesetzt. ProM ist ein erweiterbares Framework, das eine Vielzahl von Prozess Mining Techniken in Form von Plug-Ins unterstützt. Es ist plattformunabhängig (in Java) implementiert und kann kostenlos verwendet

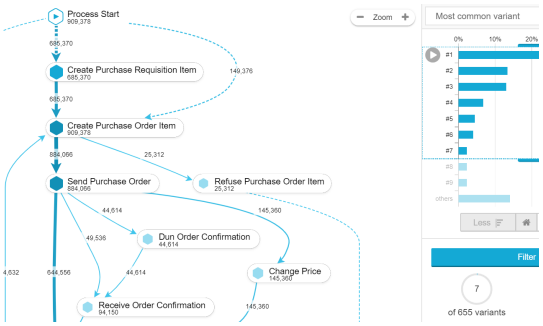

Abbildung 8: Beispiel von Prozessvarianten für einen Einkaufsprozess in Celonis - basierend auf Celonis' Demodaten

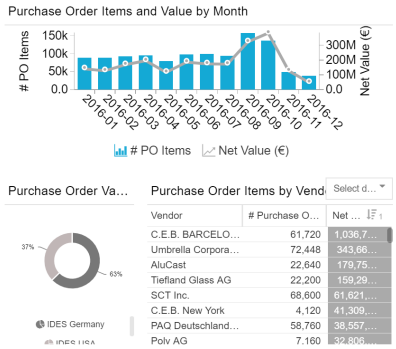

Abbildung 9: Beispiel von Kennzahlen für einen Einkaufsprozess in Celonis - basierend auf Celonis' Demodaten

werden. Anwendung findet ProM hauptsächlich in wissenschaftlichen Bereichen. Die Algorithmen gehen über eine reine Visualisierung der Prozesse/ deren Varianten hinaus - es besteht beispielsweise die Möglichkeit, ein konkretes Petri-Netz aus den Log-Files abzuleiten, wie dies in Abbildung zu erkennen ist. Hier werden demnach durchaus erweiterte Algorithmen (etwa wie der $\alpha - Algorithmus$) implementiert, für diese Arbeit ist jedoch die Ableitung des Geschäftsprozesses nicht notwendig, es genügt die Varianten anzuzeigen. Zudem ist das Tool nicht in eine S/4 HANA-Umgebung integriert.

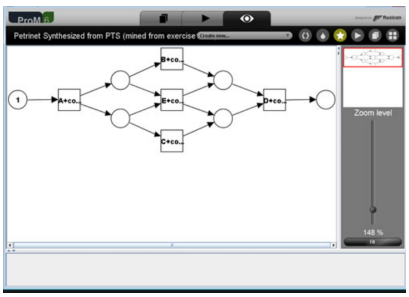

Abbildung 10: Beispiel eines Petri-Netzes - gemined in ProM (Bild aus www.promtools.org)

Disco

Das Tool „Disco“ kann verwendet werden, um Daten aus Flat-Files bzw. in Form von Tabellenkalkulationen zu verarbeiten. Die Daten müssen dazu in der Struktur eines Event-Logs vorliegen. Der wesentlicher Vorteil von Disco besteht darin, dass alle Daten die in Form eines Event-Logs vorliegen verarbeitet werden können. So können diese Daten ihren Ursprung in betriebswirtschaftlichen Informationssystemen haben oder aus Server Log-Files bestehen. Der von Disco verwendete Prozess Mining Algorithmus kann als eine verbesserte und weiterentwickelte Version des Fuzzy Miner aus ProM verstanden werden (vgl. [1]). Dieser Algorithmus erlaubt zum Beispiel die Wiedergabe eines Event-Logs basierend auf der ausgewählten Abstraktionsebene, was zu einer besseren Skalierbarkeit und Robustheit von Disco beiträgt. Im Sinne der Arbeit kann der Workflow bei diesem Tool jedoch nicht als echtzeitbasiert gesehen werden und es ist nicht in S/4 integriert. Die Abb. 11 zeigt den zuvor extrahierten Einkaufsprozess aus SAP S/4 HANA in Disco dargestellt. Die extrahierten Prozesse werden als Varianten mit Häufigkeiten dargestellt, anders als in ProM möglich, werden die Prozesslogiken nicht extrahiert.

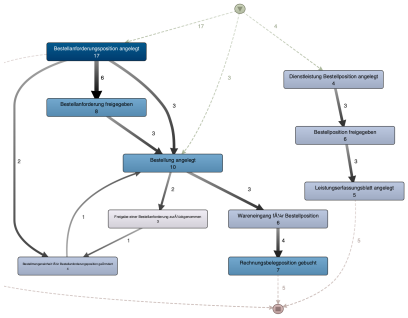

Abbildung 11: Einkaufsprozess dargestellt in Disco, basierend auf einem Event-Log aus S/4 HANA

SAP Process Observer

Der SAP Process Observer wurde mit dem Ziel entwickelt, Geschäftsprozesse über verschiedene Anwendungen hinweg über ihren gesamten Lebenszyklus des Prozessmanagements zu verwalten und zu optimieren [10]. Vor der Verwendung des SAP Process Observer, müssen im Customizing Geschäftsprozessdefinitionen, Geschäftsobjekte und Aufgabendefinitionen gepflegt werden. Durch die Geschäftsprozessdefinition wird das Customizing des Geschäftsprozesses in einen Rahmen gekapselt, so dass Geschäftsobjekten innerhalb der definierte Aufgaben zugeordnet werden können, welche wiederum die auszuführenden Aktivitäten beinhalten. Der Process Observer bietet die Möglichkeit der Zuordnung zwischen Aufgaben und Aktivitäten, um zwischen Aktivitäten, die auf Positionsebene ausgeführt werden, zu unterschieden. Somit kann die Event-Log Aufzeichnung erst nach einem durchgeführten Customizing erfolgen und nicht auf Bestandsdaten angewendet werden! Im Customizing des SAP Process Observer werden anschließend die zuvor definierten Geschäftsobjekte, Aufgaben und Aktivitäten in zusammengesetzter Form entsprechenden Events zugeordnet. Dies geschieht zur Unterscheidung von Transaktionen die für das Event bzw. für die Ausführung des Geschäftsprozesses verantwortlich sind.

Der Process Observer ist somit auch ein Tool, das Kunden bei der Analyse ihrer Prozesse unterstützten kann. Durch seine Erweiterbarkeit ist dies auch über Systemgrenzen hinweg möglich, so dass durch entsprechendes Customizing nahezu jeder Prozess integriert werden kann. Oftmals bestehen die zu untersuchenden Daten aber bereits vorher in einem Anwendungssystem und sollen zur Optimierung bestehender Prozesse analysiert werden. Primärziel des Process Observer ist es, Event-Log Daten aufzuzeichnen, die anschließend einer nachgelagerten Analyse, zum Beispiel in ProM, Anwendung finden können. Es besteht zwar die Analyse der erzeugten Event-Log Daten, jedoch können diese nicht konzentriert in ein einheitliches Prozessmodell überführt werden. Innerhalb eines Systems, das auf dem NetWeaver AS ABAP basiert, geschieht die Analyse mit Hilfe von Reports. Zur nachgelagerten Analyse liefert die SAP einen vordefinierten „Business Intelligence Content for SAP Process Observer“ aus, um die Prozess Performance im Business Warehouse überwachen zu können.

Zusammenfassender Vergleich der Mining Tools

Bezüglich des ersten Merkmals (Art des Modells) scheinen Celonis und Disco sich stark zu ähneln: es werden Übergänge ohne Semantik dargestellt; also ein Low-Level Modell erzeugt, welches sämtliche Übergänge aller Instanzen aggregiert darstellt - so, wie es auch für das zu implementierende Tool angestrebt wird. ProM ist zumindest mit entsprechenden Plug-Ins in der Lage High-Level-Modelle mit Semantik darzustellen; der Process Observer liefert zuerst ein informelles Modell, welches natürlich mit entsprechenden BI-Tools dann weiterverarbeitet werden kann. Bezüglich des zweiten Merkmals sind zumindest ProM und Disco klar als nicht echtzeitbasiert zu identifizieren, da die Auswertung einen vorherigen Dateidownload erfordert. Aber auch der Process Observer liest eigentlich nur bereits vorher aufgezeichnete historische Logs aus. In Celonis hängt es letztlich davon ab, wie eine Schnittstelle in das Anwendungssystem implementiert ist, was aber auch nicht Teil des S/4 HANA Standards darstellt. Bezüglich des dritten Merkmals ist festzuhalten, dass kein Tool direkt in S/4 HANA mit Hilfe von UI5 eine Auswertung der Miningergebnisse erzeugt.

Einkaufsprozesse in S/4 HANA

Sind die Event-Logs bereits extrahiert, spielt die Auswahl eines spezifischen Geschäftsprozesses eigentlich keine Rolle mehr. Da eine vorherige Extraktion der Logs zumindest bei einem Echtzeittool aber einen direkten Zugriff auf die aktuellen Datenbanktabellen bedingt, sind für die betrachteten Prozesse im Vorfeld festzulegen. Es wurde in dieser Arbeit entschieden, zwei grundsätzlich verschiedene Einkaufsprozesse zu untersuchen:

- Einkauf von indirekten Materialien
- Einkauf von Dienstleistungen

Der jeweilige Referenzprozess für jeden der beiden Prozesse wird in Abbildung 12 und 13 dargestellt. Der wesentliche Unterschied zwischen beiden Prozessen besteht in der Art der Leistungserbringung, auch wenn in der Darstellung in Abbildung 13 keine Anforderungsphase dem Bestellprozess vorausgeht. Sie stammen aus dem „SAP Best Practices Explorer“ und können als typisch für diese verschiedenen Arten der Beschaffung angesehen werden, wobei die Frage, wie ge-

nerisch diese Prozesse nun wirklich sind, nicht relevant ist, so lange die Prozessdaten entsprechend von den Datenbanken extrahierbar sind und in ein Event-Log umgewandelt werden können.

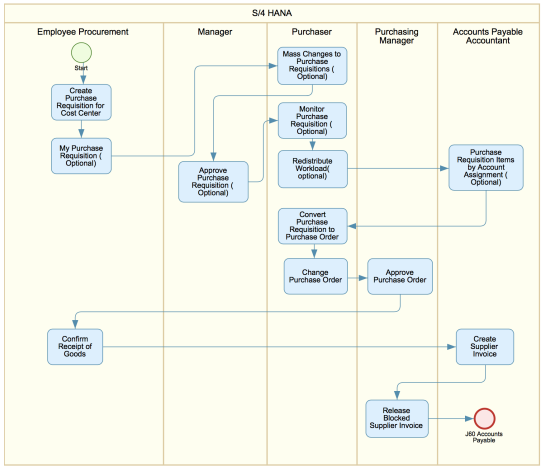

Abbildung 12: Beschaffung indirekter Materialien [11]

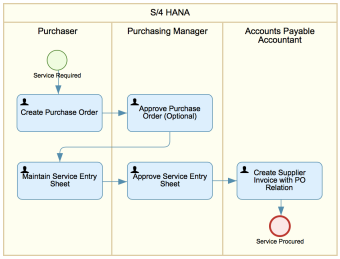

Abbildung 13: Beschaffung von Dienstleistungen [12]

Extraktion von Einkaufsinformation

Im Rahmen der Extraktion der Prozessdaten müssen für die ausgewählten Einkaufsprozesse die folgenden Daten zur Repräsentation des Prozesses im Event-Log ausgewählt werden:

1. Bestellanforderungspositionen
2. Bestellpositionen und Leistungserfassungsblättern
3. Wareneingänge
4. Rechnungseingänge

Das Event-Log muss auf Positionsebene (nicht Kopfebene) aufgebaut werden, da für einzelne Positionen unterschiedliche Aktivitäten ausgeführt sein können. Bestellanforderungen oder (typischer Bestellungen) können zum Beispiel Positionen enthalten, die aus anderen Einkaufsobjekten, wie etwa Bestellanforderungen, kopiert wurden, oder eine Position einer Bestellanforderung wurde bereits als erledigt gekennzeichnet. Es wird hier die Extraktion der Bestellanforderungspositionen beschrieben; die Extraktion der übrigen Positionen wurden zwar im Prototypen ebenfalls durchgeführt, aber die Logik ist analog zur den Bestellanforderungspositionen. Ausgangspunkt für die Erstellung eigener CDS-Views ist der Standardview „I_PURCHASEREQUISITIONITEM“. Ein Event-Log besteht aus einer Sammlung von Ereignisdaten. Führendes Element ist die Prozessinstanz, der die Aktivitäten und Zeitstempel folgen. Um die Aktivitäten einer Prozessinstanz nachvollziehbar zuordnen zu können, ist sicherzustellen, dass für jeden Prozess eine einzigartige Prozessinstanz erzeugt wird. Da in einem SAP-System der Mandant Primärschlüssel der Applikationstabellen ist (jedoch nicht der CDS-View) trennt der Mandant die Daten der verschiedenen Mandanten und muss somit auch Teil des Extraktes sein. Weitere Daten sind dann die „Belegnummer“ und die „Belegposition“ verbunden mit einem „Timestamp“ und einer „Aktivität“.

Das Feld „Erstellungskennzeichen“ dokumentiert die Herkunft einer Bestellanforderungsposition und spielt deshalb bei der Exktraktion der Positionen eine Rolle - in diesem Prozess sind drei Herkunftsarten relevant:

1. **D** - Direktbeschaffung
2. **R** - Realtime (manuell)
3. **S** - Beschaffung per Self Service

Für die zu extrahierende Aktivität ist zudem im Detail zu extrahieren, was konkret auf der Bestellanforderungsposition erfolgt ist; mögliche Aktivitäten sind dann:

1. **Anlage** - Bestellanforderung angelegt
2. **Löschung** - Bestellanforderung gelöscht
3. **Änderung** - Bestellanforderung geändert
4. **Freigabe** - Bestellanforderung freigegeben
5. **Löschung Rücknahme** - Löschung einer Bestellanforderung zurückgenommen
6. **Freigabe Rücknahme** - Freigabe einer Bestellanforderung zurückgenommen

Für die Aktivität „Bestellanforderung angelegt“ sind sämtliche Informationen im selektierten CDS-View bereits vorhanden, maßgeblich ist hier das Anlagedatum (CreationDate) der CDS-View. Die Bestimmung eines exakten Zeitstempels ist aufgrund fehlender Daten für Bestellanforderungspositionen nicht möglich. Daher muss für Bestellanforderungspositionen die Annahme getroffen werden, dass der Zeitstempel einfach auf 00:00:00 Uhr definiert werden kann. Alle weiteren Aktivitäten können über den CDS-View Änderungsbelegpositionen „I_CHANGEDOCUMENTITEM“, welche zusammen mit „I_CHANGEDOCUMENT“ zu lesen ist, identifiziert werden. Wie zu erkennen ist, sind eine Reihe von Standard-CDS-Views betroffen, die zur Abbildung der Selektionslogik wiederum in eigene CDS-Views eingebaut werden. Die endgültige Consumption-View für die Extraktion ist dann in Listing 4 dargestellt, ein Beispiel einer Interface-View, die darin eingeht, stellt das Listing 3 dar. In diesem Listing werden die entsprechenden Änderungsbelege in Kombination mit den Bestellanforderungspositionen gelesen um die relevanten Änderungen zu selektieren. Dies geht dann in die View zur Selektion aller Aktivitäten der Bestellanforderungspositionen ein.

Listing 3: „Interface“ CDS-View „Z_I_PURREQUITEMACTIVITYSC“ zur Selektion von Bestellanforderungspositionen mit dazugehörigen Änderungsbelegpositionen.

```
1 @AbapCatalog.sqlViewName: 'ZIPMPURREQACTSC'
```

```
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Event Log: Positionen fuer Bestellanforderung'
@VDM.viewType: #BASIC
define view Z_I_PURREQUITEMACTIVITYSC as select from
      Z_I_PurchaseRequisitionItem
  association [1..1] to I_ChangeDocumentItem as _ChangeDocumentItem on
       $projection.PurchaseRequisition          = _ChangeDocumentItem.
       ChangeDocObject
       and _ChangeDocumentItem.ChangeDocObjectClass = 'BANF'
  association [1..1] to I_ChangeDocument as _ChangeDocument on $projection.
       changedocument = _ChangeDocument.ChangeDocument
       and _ChangeDocument.ChangeDocObject = PurchaseRequisition
{
  key   PurchaseRequisition,
  key   PurchaseRequisitionItem,
        PurReqnRequestor                                as Requestor,
        _ChangeDocumentItem.ChangeDocument,
        _ChangeDocumentItem.ChangeDocDatabaseTableField,
        _ChangeDocumentItem.ChangeDocNewFieldValue,
        _ChangeDocumentItem.ChangeDocPreviousFieldValue as
              ChangeDocOldFieldValue,
        /* Associations */
        _ChangeDocument
}
```

Listing 4: „Consumption“ CDS-View „Z_C_PURREQUISITIONITEMACTIVITY“ zur Selektion von Bestellanforderungspositionen.

```
@AbapCatalog.sqlViewName: 'ZCPMPURREQITMACT'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Event-Log: Positionen fuer Bestellanforderung'
@VDM.viewType: #CONSUMPTION
define view Z_C_PURREQUISITIONITEMACTIVITY as select from
      Z_I_PurchaseRequisitionItem
{
 key concat(concat(concat(concat(mandt, '-'), PurchaseRequisition),'-'),
      PurchaseRequisitionItem) as CaseId,
      concat(rtrim(CreationDate, '-'), '000000') as Timestamp,
      case $session.system_language
          when 'E' then 'Create Purchase Requisition Item'
          when 'D' then 'Bestellanforderungsposition angelegt'
          else 'No activity maintained for this logon language'
      end as Activity,
      PurReqnRequestor as Requestor
}

union all

select from Z\_I\_PURREQUISITIONITEMACTIVITY
{
    key concat(concat(concat(concat(mandt, '-'), PurchaseRequisition),'-'),
        PurchaseRequisitionItem) as CaseId,
        concat(_ChangeDocument.CreationDate, _ChangeDocument.CreationTime)
            as Timestamp,
        _Activity.ActivityDescription as Activity,
        Requestor
} where _Activity.ActivityDescription <> ''

union all

select from Z\_I\_PURREQUITEMACTIVITYSC2
{
  key concat(concat(concat(concat(mandt, '-'), PurchaseRequisition),'-'),
      PurchaseRequisitionItem) as CaseId,
      concat(CreationDate, CreationTime) as Timestamp,
      ActivityDescription as Activity,
      Requestor
}
```

Es sind somit pro ausgewählter Position des Einkaufsprozesses eine Reihe von CDS-Views zu erstellen, die eine Hierarchie basierend auf Standard-Views bilden, um die Extraktion abzubilden. Ein Beispiel für ein komplettes Log, welches auch wiederum über eine CDS-View implementiert wurde, zeigt dann Abbildung 14.

Darstellung von Prozessen in S/4 HANA

SAP UI5 Network Graph

Der „SAP UI5 Network Graph“ dient zur Darstellung großer Datenmengen, bei denen die Beziehungen zwischen den einzelnen Datensätzen hervorgehoben werden [13]. Datensätze werden als Knoten angezeigt, und Konnektoren stellen die Beziehungen zwischen ihnen dar. Ein einfaches Beispiel des Network-Graphen mit drei Knoten und zwei Konnektoren zeigt Abbildung 15.

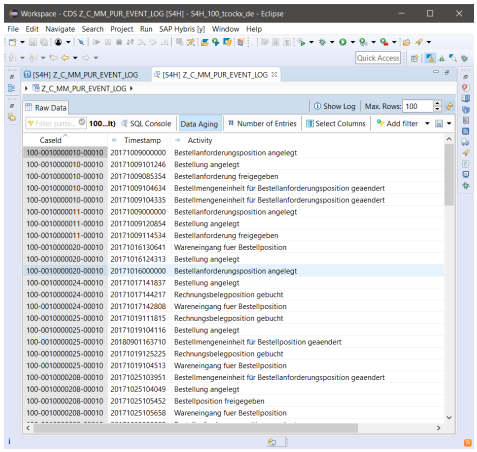

Abbildung 14: Vorschau auf CDS-View zur Abbildung des Event-Logs

Abbildung 15: Elemente des Network Graphen

Die Knoten und Konnektoren werden innerhalb des „Network Graph“ im sogenannten „Network Graph Content“ Bereich dargestellt. Ergänzt wird dieser Bereich um eine „Network Graph Toolbar“ und eine „Network Graph Map“; diese stellt den gesamten Graphen in einer verkleinerten Navigationssicht dar (Abbildung 16) - das Konstrukt erlaubt somit eine Navigation über größere Netzwerke. Zur verbesserten Übersicht und Lesbarkeit sind für den „Network Graph“ Start- und Endknoten eingeführt worden. Diese Knoten werden im Backend grundsätzlich erzeugt, so dass für den Anwender immer ersichtlich ist, wo der Prozess begonnen hat und mit welchen Knoten der Prozess beendet wird. Der „Network Graph“ unterstützt im Standard nur die Darstellung von großen Netzwerken, für die anschauliche Darstellung von Netzwerkknoten, welche nicht triviale Datenunterschiede aufzeigen, die bisher übersehen worden wären [13].

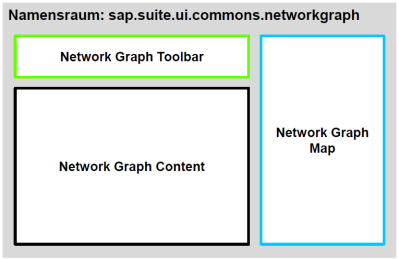

Abbildung 16: Schematischer Aufbau des Network Graphen

Aggregationskonzepte in SAP UI5

In SAP UI5 erben alle Controls von der Oberklasse „sap.ui.core.Control“ (alles aus [14]), diese wiederum erbt von der abstrakten Superklasse „sap.ui.base.Object“. Ausgehend vom „Network Graph“ ist dies auch der Fall, so dass alle Con-

trols im Namensraum „sap.suite.ui.commons.networkgraph“ des „Network Graph“ auf der Klasse „sap.ui.core.Control“ basieren. Controls können andere Controls aggregieren, sofern die Angabe von Aggregationen bei einem Control zulässig sind. Das Control dient dann als Container, dem die Applikation sogenannte untergeordnete Controls hinzufügen kann. In SAP UI5 wird dann von einem „zusammengesetzten Control“ gesprochen, wenn das Control selbst untergeordnete Controls hinzufügt und verfügbare Komponenten wiederverwendet.

Aggregationen eines Controls beschreiben also eine Sammlung in der alles definiert werden kann, was vom übergeordneten Control vorgesehen ist. Zur Laufzeit sind untergeordnete Controls im Besitz des übergeordneten Control und werden zusammen mit dem übergeordneten Control vernichtet. In Abbildung 17 ist ein Auszug der möglichen Aggregation, die für den „Network Graph“ in Frage kommen zusammengefasst.

Name	Kardinalität	Klasse	Beschreibung
lines	[0..n]	sap.suite.ui.commons .networkgraph.Line	Linien die im Diagramm dargestellt werden sollen.
nodes	[0..n]	sap.suite.ui.commons .networkgraph.Node	Knoten die im Diagramm dargestellt werden sollen.
groups	[0..n]	sap.suite.ui.commons .networkgraph.Group	Enthält eine Liste der im Diagramm verwendeten Gruppen.

Abbildung 17: Aggregationen für den Network-Graphen

Aggregations Bindung ist als Erweiterung des Konzepts zu verstehen. Verwendung findet es, um zum Beispiel basierend auf Backend Daten durch Klonen eines Template Controls oder durch die Verwendung einer Factory-Funktion untergeordnete Controls zu erzeugen. Am Beispiel des „Network Graph“ werden solange Knoten im Diagramm erzeugt, wie vom Backend Daten durchgereicht werden. Die Aggregations Bindung kann entweder direkt im Konstruktor des übergeordneten Controls angegeben werden oder durch den Aufruf der Methode „bindAggregation“ des Control Objekts.

Datenbereitstellung

Das zuvor vorgestellte Konzept der Aggregations Bindung hat zur Folge, dass sich der bereitzustellende SAP Gateway Service aus der Aufrufstruktur des Network Graph ableitet. Um das Prozessmodell darstellen zu können, werden Aggregationen für Prozessknoten und Konnektoren zwischen diesen benötigt. Zur Bereitstellung der Daten verwendet das SAP Gateway das Open Data Protocol (kurz OData). Das SAP Gateway ist integraler Bestandteil des SAP NetWeaver Anwendungsservers und ermöglicht die Anbindung von Geräten, Umgebungen und Plattformen an SAP Systeme [2]. Dadurch kann jede beliebige Programmiersprache oder jedes beliebige Modell verwendet werden, um eine Verbindung zwischen SAP und nicht SAP Anwendungen herzustellen. Das OData Protokoll ermöglicht die Erstellung von HTTP-basierten Datendiensten, die Ressourcen werden mit Hilfe von Uniform Resource Identifiers (kurz URIs) identifiziert und in einem abstrakten Datenmodell definiert. Die Definition des abstrakten Datenmodells dient dann zur Implementierung, um die erforderlichen Daten bereitzustellen. In Abb. 18 ist ein Auszug aus dem im Zuge dieser Arbeit entwickelten Gateway Service zu sehen. Die Entitätsmenge „GraphNodes“ dient im Anschluss als URI, um zum Beispiel die erforderlichen Daten zur Erstellung der Prozessknoten zu erhalten. Die Abbildung 20 zeigt die XML-View zur Erzeugung des Network Graph aus der SAP Web IDE unter Verwendung der Aggregationen „nodes“ und „lines“, Abbildung 19 zeigt ein Standardbeispiel des Network-Graphen ohne die Mining-Prozessspezifischen Elemente.

Implementierung von SAP Gateway Services

Es bestehen mehrere Möglichkeiten der Implementierung von SAP Gateway Services. Zur Eingrenzung werden ausschließlich die beiden Ansätze, die in dieser Arbeit verwendet werden, vorgestellt. Zur Umsetzung der analytischen KPIs und Diagramme werden CDS-Views zur Datenextraktion modelliert, welche auf Basis der zuvor erstellten „Interface Views“ im Rahmen der Datenextraktion entstanden sind. Dies ist dem Grund geschuldet, dass ausschließlich Datensätze in die Betrachtung mit einfließen sollen, die auch zur Erzeugung des Prozessmodells Verwendung finden. CDS-Views, deren einziger Verwendungszweck es ist, auf Daten aus Datenbanktabellen zuzugreifen und diese über „Consumption Views“ für individuelle Konsumenten des VDM bereitzustellen, werden am besten über eine Datenquellenreferenz bereitgestellt. Eine Umsetzung dieser Datenquellenreferenzen erfolgt mit Hilfe der Service Adaptation Definition Language (SADL), welche dann als Infrastruktur für das modellbasierte Lesen und Verarbeiten von Daten dient. Das abgeleitete Datenmodell aus der referenzierten CDS-View dient somit wiederum der Definition des abstrakten Datenmodells des zu konsumierenden SAP Gateway Services und wird zusätzlich zur Festlegung der Namen der einzelnen URIs verwendet. Je spezifischer die Anforderungen an einen SAP Gateway Services sind, umso mehr empfiehlt sich eine eigene Implementierung über die Programmiersprache ABAP. Die im Zuge dieser Arbeit entwickelten globalen ABAP Klassen, in denen der Kontrollflussalgorithmus implementiert ist, werden zur Bereitstellung der Daten im SAP Gateway Service konsumiert. Eine Auslagerung des Algorithmus in globale ABAP Klassen ermöglicht dann auch eine potentiell zukünftige Implementierung fortgeschrittener Algorithmen.

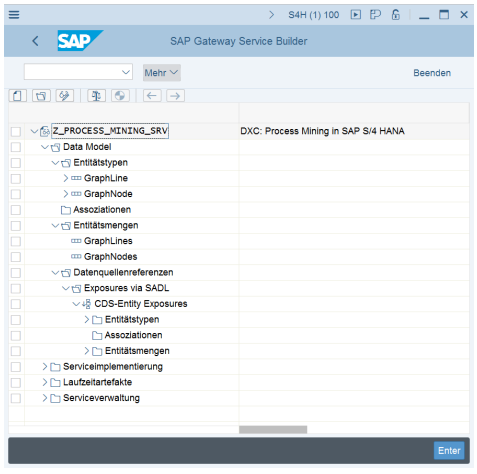

Abbildung 18: Entwickelter SAP Gateway Service im SAP Gateway Service Builder (Transaktion SEGW).

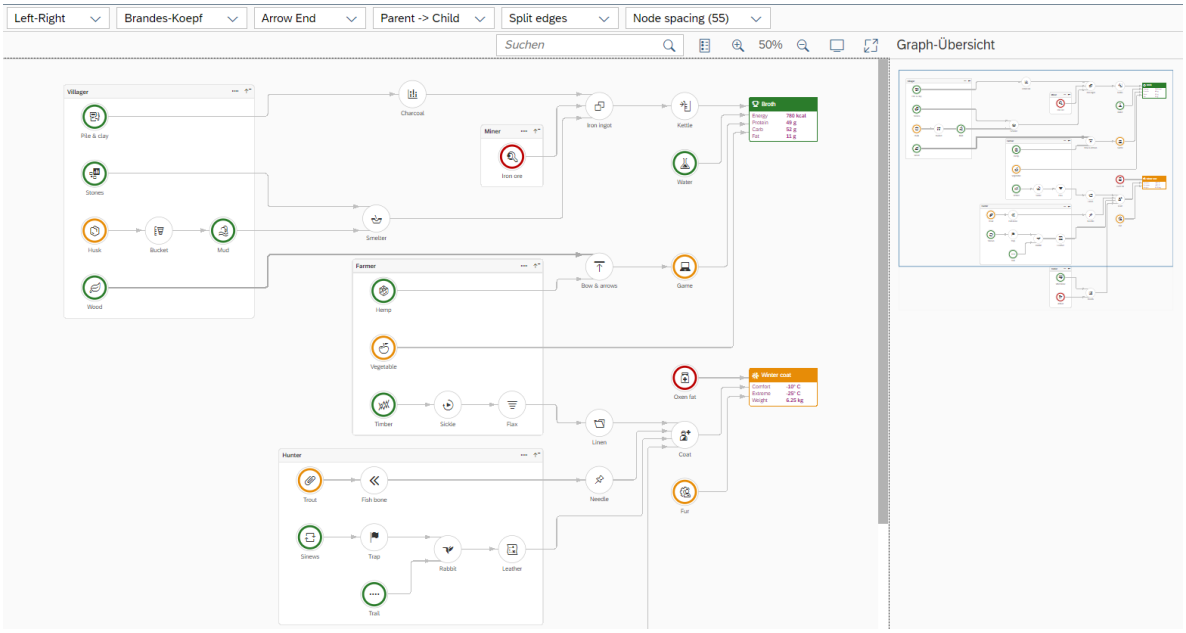

Abbildung 19: Network-Graph aus den SAP UI5 Beispielen.

```
<l:flexContent>
    <m:FlexBox fitContainer="true" renderType="Bare" wrap="Wrap" id="graphWrapper">
        <m:items>
            <Graph id="graph" nodes="{/GraphNodes}" lines="{/GraphLines}" enableWheelZoom="false" orientation="{settings>/orientation}">
                <layoutData>
                    <m:FlexItemData growFactor="1" shrinkFactor="1" baseSize="0%" minWidth="400px"/>
                </layoutData>
                <layoutAlgorithm>
                    <layout:LayeredLayout mergeEdges="{settings>/mergeEdges}" nodePlacement="{settings>/nodePlacement}" nodeSpacing="{settings>/nodeSpacing}"/>
                </layoutAlgorithm>
                <nodes>
                    <Node key="{Key}" title="{Title}" icon="{Icon}" group="{group}" shape="{Shape}" status="{status}" width="300px">
                        <attributes>
                            <ElementAttribute label="{i18n>processTaskCount}" value="{Count}"/>
                        </attributes>
                    </Node>
                </nodes>
                <lines>
                    <Line arrowPosition="End" arrowOrientation="ParentOf" lineType="solid" from="{From}" to="{To}" attachHover="displayoverPopUp">
                        <attributes>
                            <ElementAttribute label="{i18n>processTaskCount}" value="{Count}"/>
                        </attributes>
                    </Line>
                </lines>
            </Graph>
            <GraphMap id="map" graph="graph">
                <layoutData>
                    <m:FlexItemData minWidth="200px" maxWidth="25%"/>
                </layoutData>
            </GraphMap>
        </m:items>
    </m:FlexBox>
</l:flexContent>
```

Abbildung 20: XML-View zur Erzeugung des Prototypen des Network-Graphen

PROTOTYP EINES PROZESS MINING TOOLS

Eine Implementierung der Controls und Füllen mit den Extraktionsdaten sowie einer Berechnung der Kennzahlen führt dann zu dem Prototypen des Mining-Tools, welches hier mit Hilfe einiger Screenshots dokumentiert werden soll. Wie zu erkennen ist, kann rechts zwischen einer reinen Navigation oder auch zwischen einer Navigation, kombiniert mit einer Anzahl von Kennzahlen ausgewählt werden. Die Knoten wurden mit den entsprechenden Icons versehen.

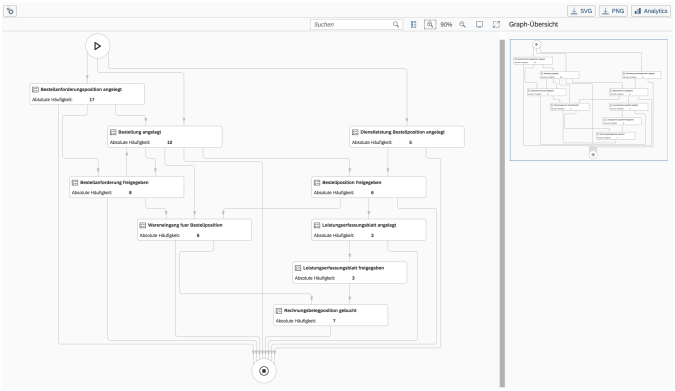

Abbildung 21: Prozessmodell auf Basis des Network Graph

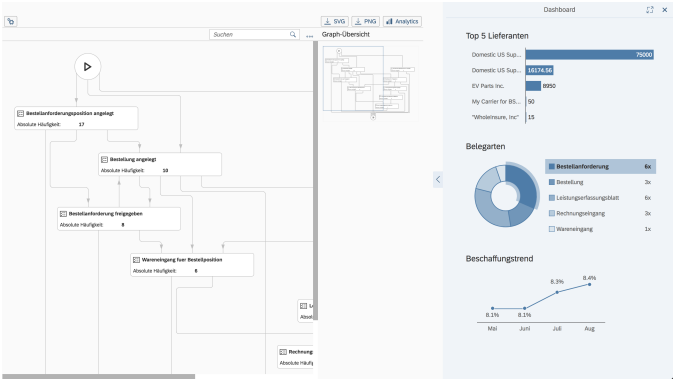

Abbildung 22: Networkgraph angereichert mit Kennzahlen

Das erstellte Prozessmodell ist in der Lage, ein in der Übergangsmatrix dargestellte zuvor extrahierte Event-Log mit Hilfe des „Network Graph“ wiederzugeben und die verschiedenen Traces darzustellen. Fortgeschrittenere Prozess Mining Tools sind oftmals in der Lage, auf Basis von Prozessinstanzen, Aktivitäten und Zeitstempel zusätzliche Informationen zu extrahieren. So ist zum Beispiel die Erweiterung denkbar, den „Network Graph“ dahingehend anzupassen, dass die Dicke der Konnektoren in Abhängigkeit der absoluten Häufigkeit aufgetretener Übergänge bestimmt wird. Auf Basis der Zeitstempel könnte die absolute Gesamtdauer zwischen zwei Prozessknoten bestimmt werden, sinnvollerweise könnte über eine Wiedergabe der Dauer über den Median ebenfalls nachgedacht werden. Zwischen Prozessknoten wird üblicherweise die Anzahl der Übergänge anhand von Kantenbeschriftungen der Konnektoren visuell dargestellt. Der „Network Graph“ bietet diese Möglichkeit nicht im Standard, kann aber die Anzahl dieser Übergänge anhand von Detailfenstern darstellen. Diese Detailfenster können üblicherweise für jeden Konnektor innerhalb des Prozessmodells aufgerufen werden. So wäre die Folgeimplementierung denkbar, die zuvor beschriebenen Werte der Gesamtdauer zwischen zwei Prozessknoten in diesem Fenster mit aufzunehmen. Dies erfordert eine entsprechende Berücksichtigung bei der Ermittlung des Prozessmodells und eine Anpassung bei der Datenbereitstellung.

RESUME UND AUSBLICK

In dieser Arbeit sollte die technischen Machbarkeit einer Implementierung eines echtzeitbasierten Prozess Mining Tools in S/4 HANA untersucht werden. Es bestanden die Hauptprobleme in der Datenextraktion und Visualisierung des Prozessmodells. Der entwickelte Ansatz zeigt, dass im Zuge der Datenextraktion weitgreifende Möglichkeiten bestehen, so stellt die CDS-Datenmodellierung alle notwendigen Funktionen der Erzeugung von Event-Logs bereit. Aufgrund der In-Memory-Technologie von S/4 HANA waren die Daten auch effizient extrahierbar. Es ist jedoch zu bemerken, dass eine Reihe von CDS-Views erstellt werden mussten, um das Event-Log für den Einkaufsprozess zu erhalten. Hier sollte in Zukunft ein generischeres Konzept der Extraktion stehen, um zu vermeiden, dass eine Vielzahl solcher Views anzulegen ist. Der in dieser Arbeit verwendete Algorithmus zur Erzeugung des Prozessmodells, basiert auf Grundideen der Literatur (speziell [9]) und könnte zukünftig durch fortschrittlichere Prozessmining-Funktionen ersetzt werden. Es konnte gezeigt werden, dass sich ein bestehendes Control des SAP UI5 Frameworks in ein informelles Prozessmodell überführen lässt und auf Basis der CDS Views analytische Diagramme und KPIs erstellt werden können. Der in dieser Arbeit betrachtete Geschäftsprozess konnte im Sinne der Typen von Prozess Mining entdeckt werden und das in Echtzeit zur Beantwortung von wiederkehrenden Routinefragen. Es gilt jedoch zu prüfen, wie sich die Antwortzeiten verhalten, sobald 10.000 oder gar 1 Million Datensätze im System entdeckt werden sollen; dies konnte hier nicht geprüft werden, es ist jedoch zu bemerken, dass mit dem jetzigen Datenbestand eine Echtzeitverarbeitung möglich war und dass die Daten alle direkt von der In-Memory Datenbank extrahiert werden, was vermuten lässt, dass die Laufzeit bei der Extraktion auch größerer Datensätze noch in Echtzeit erfolgen kann - dies ist jedoch zu validieren.

LITERATUR

[1] Wil van der Alst. *Process Mining Data Science in Action, Second Edition.* Springer Verlag, 2016.

[2] SAP. Sap gateway foundation developer guide. https://help.sap.com/de/7.5.12/en-US/SAPGWDeveloperGuide.htm. Zuletzt eingesehen am 12.10.2018.

[3] Hubert Oesterle. *Business Engineering. Prozeß- und Systementwicklung.* Springer-Verlag, 1995.

[4] Eindhoven University of Technology. Process Mining Group, Math & CS department. http://www.processmining.org/event_logs_and_models_used_in_book, 2016. Zuletzt eingesehen am 12.10.2018.

[5] SAP. ABAP Programming Model for SAP Fiori. https://help.sap.com/viewer/-cc0c305d2fab47bd808adcad3ca7ee9d/7.51.2/en-

US/3b77569ca8ee4226bdab4fcebd6f6ea6.html. Zuletzt eingesehen am 12.10.2018.

[6] Abani Pattanayak. Sap s4hana embedded analytics: An overview. https://file.scirp.org/pdf/JCC_2017063016573251.pdf, 2017. Zuletzt eingesehen am 12.10.2018.

[7] Colle and Dentzer et al. *Core Data Services für ABAP*. Rheinwerk Verlag, 2018.

[8] SAP. Expose cds view as an odata service. https://help.sap.com/viewer/-cc0c305d2fab47bd808adcad3ca7ee9d/7.5.9/en-US/79cb3bf4eafd4af9b39bc6842e5be8bd.html, 2018. Zuletzt eingesehen am 12.10.2018.

[9] Diogo R. Ferreira. *A Primer on Process Mining: Practical Skills with Python and Graphviz*. SpringerBriefs in Information Systems. Springer, 2017.

[10] SAP. Sap process observer. https://help.sap.com/erp2005_ehp_06/help-data/en/33/14dd25b1964c6b8b44cf5b9d757b81/frameset.htm. Zuletzt eingesehen am 12.10.2018.

[11] SAP. Sap best practices explorer - requisitioning. https://rapid.sap.com/bp/#/browse/categories/sap_s4hana/areas/on-premise/packageversions/BP_OP_ENTPR/S4HANA/1709-/DE/4/EN/scopeitems/18J, 2017. Zuletzt eingesehen am 12.10.2018.

[12] SAP. Sap best practices explorer - procurement of services. https://rapid.sap.com/bp/#/browse/categories/sap_s4hana/areas/on-premise/packageversions/BP_OP_ENTPR/S4HANA/1709-/DE/4/EN/scopeitems/22Z, 2017. Zuletzt eingesehen am 12.10.2018.

[13] SAP. Sap fiori design guidelines network graph. https://experience.sap.com/fiori-design-web/network-graph/. Zuletzt eingesehen am 12.10.2018.

[14] SAP. Open ui5 hana on demand. https://openui5.hana.ondemand.com/. Zuletzt eingesehen am 12.10.2018.