HAW HAMBURG

Masterarbeit

David Schwärzler

Identifikation strukturgebender Felder in Eventlogs

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

David Schwärzler

Identification of key process identifiers in event logs

Master Thesis based on the examination and study regulations
for the Bachelor of Engineering degree programme
Master of Science Informatik
at the Department of Information and Electrical Engineering
of the Faculty of Engineering and Computer Science
of the University of Applied Sciences Hamburg
Supervising examiner: Prof. Dr. Ulrike Steffens
Second examiner: Prof. Dr. Martin Schultz

Day of delivery: 12.12.2024

David Schwärzler

Thema der Arbeit

Identifikation strukturgebender Felder in Eventlogs

Stichworte

Process-Mining, Process-Discovery, Eventloganalyse

Kurzzusammenfassung

Herkömmliche Prozess-Mining-Verfahren erfordern strukturierte Eventlogs, in denen die Fall- und Aktivitäts-Ids eindeutig definiert sind. In vielen realen Szenarien, insbesondere bei Eventlogs aus Produktivsystemen, ist diese Voraussetzung jedoch nicht immer erfüllt. Diese Arbeit stellt ein Verfahren vor, mit dem die Fall- und Aktivitäts-Ids in unstrukturierten oder unvollständigen Eventlogs automatisch erkannt und extrahiert werden können.

David Schwärzler

Title of Thesis

Identification of key process identifiers in event logs

Keywords

Process mining, Process discovery, Event log analysis

Abstract

Conventional process mining techniques require structured event logs where case and activity Ids are clearly defined. However, this condition is not always met in real life scenarios. This thesis presents a method for the automatic identification and extraction of case and activity Ids in such unstructured or incomplete event

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

Glossar

das intelligente System Die Komponente des Prototyps, die mittels maschinellem Lernen die Kennzahlen auswertet und die korrekte Fall- und Aktivitäte-Id prognostiziert.

Falsches Kandidatenpaar Das dalsche Kandidatenpaar ist ein Kandidatenpaar, dass nicht die tatsächlichen Felder für fall und Aktivitäts-Id enthält. Ein Kandidatenpaar ist auch falsch, wenn nur ein Element des Tupels von diesen abweicht.

intelligente System ist die Komponente im Prototyp, die mitels maschinellem Lernen die Kennzahlen auswertet.

Kandidatenpaar Ein Kandidatenpaar ist ein Tupel, dass aus 2 Feldern des Eventlogs besteht. Das erste Element wird als potentielle Fall-id und das zweite Element als potentielle Akivitäts-Id angesehen.

Kennzahlen Die Kennzahlen werden anhand der Charakteristiken der Muster berechnet und enthalten Metriken wie bspw. die Länge der Muster oder deren Häufigkeit des Auftretens.

Korrektes Kandidatenpaar Das korrekte Kandidatenpaar ist ein Kandidatenpaar, dass die tatsächliche Fall- und Aktivitäts-Id des Eventlogs enthält.

Muster Ein Muster ist eine Sequenzen, die mehrmals in einem Eventlog auftritt.

Process Discovery Die automatisierte Rekonstruktion der struktur eines Geschäftsprozesses anhand eines Eventlogs.

Sequenz Seuqnzen sind Aktivitätsreihenfolgen in einem Eventlog, die von einer Prozessinstanz verursacht werden.

Sprungpfade Sprungpfade sind spezielle Pfade einer Bedingun der Prozess-Engine, die mit einem Sprungknoten enden und nicht wieder mit den anderen Pfaden der Bedingung vereinigt werden.

Akronyme

BPMN Business Process Model and Notation.

BPMS Business Process Management Software.

CSV Comma-seperated values.

JSAT Java Statistical Analysis Tool.

JSON JavaScript Object Notation.

PLG Processes and Logs Generator.

SQL Structured Query Language.

XES Extensible Event Stream.

XLS Excel Spreadsheet.

XML Extensible Markup Language.

Keywords: Process-Mining, Process-Discovery, Eventloganalyse

1 Einleitung

Die meisten Informationssysteme erzeugen Eventlogs, die getätigte Aktionen und Ereignisse in einem System protokollieren. Jeder Eintrag in einem Eventlog kann mithilfe der Fall-Id genau einer Prozessinstanz und über die Aktivitäts-Id genau einem Schritt im Prozess zugeordnet werden [8].

Diese Informationen bilden die Grundlage für Data-Mining-Verfahren, um automatisiert Prozessstrukturen in Eventlogs zu erkennen (Process Discovery), die geplanten mit den tatsächlich durchgeführten Prozessen zu vergleichen (Conformance Checking) und Engpässe sowie Optimierungsmöglichkeiten zu identifizieren (Process Enhancement) [18, 8].

In der Praxis kommt es jedoch vor, dass gerade die Fall- und Aktivitäts-Id im Eventlog nicht bekannt sind [8]. Zurückzuführen ist dies auf eine manuelle Erfassung der Prozessschritte oder auf eine heterogene Systemlandschaft, in der kein zentrales Loggingsystem existiert [18]. In diesen Fällen wird von *unlabeled* Eventlogs gesprochen, welche sich nicht mit herkömmlichen Prozess-Mining-Verfahren verarbeiten lassen [18].

In dieser Arbeit wird ein Verfahren vorgestellt, das unlabeled zu labeled Eventlogs konvertiert, indem die unbekannten Felder für Fall- und Aktivitäts-Id automatisiert identifiziert werden. Dies erlaubt eine anschließende Analyse mit herkömmlichen Prozess-Mining-Verfahren auf den labeled Eventlogs.

1.1 Motivation und Zielsetzung

Unlabeled Eventlogs stellen eine zusätzliche Herausforderung im Bereich des Prozess-Minings dar. In dieser Arbeit wird von der konkreten Problemstellung ausgegangen, dass Fall- und Aktivitäts-Ids zwar im Eventlog enthalten sind, jedoch nicht bekannt ist, welche Felder diese Informationen enthalten. Dieses Szenario tritt bevorzugt in Unternehmen mit großen, heterogenen Systemlandschaften auf, bei denen Events nicht zentral gesammelt werden [24]. Auch können unlabeled Eventlogs die Folge von Altsystemen sein, deren Logs nicht auf die Verarbeitung von Prozess-Mining ausgelegt sind.

Traditionell ist die Identifikation der unbekannten Fall- und Aktivitäts-Id in einem unlabeled Eventlog ein manueller Prozess. Eine Automatisierung dieses Prozesses, durch das in dieser Arbeit vorgestellte Verfahren, bietet mehrere Vorteile:

Kosten- und Ressourcenreduktion Die manuelle Identifikation der Fall- und Aktivitäts-Ids erfordert ein tiefes Verständnis der zu analysierenden Prozesse und Systeme. Unternehmen, die sich für die Implementierung von Prozess-Mining interessieren, greifen häufig auf spezialisierte Dienstleister zurück. Das notwendige Domänenwissen über die Prozesse befindet sich jedoch typischerweise beim Auftraggeber und nicht beim Dienstleister. Gerade in Systemen in denen viele unterschiedliche unlabeled Eventlogs existieren, kann dies zu einem erhöhten Kommunikationsaufwand und potenziellen Fehlern führen. Eine automatisierte Identifikation sorgt für eine erhebliche Zeit- und Kosteneinsparung.

Neue Prozessperspektiven Neben der eigentlichen Fall- und Aktivitäts-Id können auch weitere Felder im Eventlog existieren, welche charakteristische Merkmale von Fall- und Aktivitäts-Ids aufweisen. Eine Analyse dieser Felder kann neue Einblicke in die Prozesse gewähren und bisher verborgene Informationen und Abhängigkeiten aufdecken. Dieses wird im Folgenden anhand des klassischen Beispiels einer Bestellung verdeutlicht.

Fall-Id	Aktivitäts-Id	Bestellstatus	Land	Spediteur
1	Bestellung eingegangen	Eingegangen	Deutschland	
1	Lagerbestand prüfen	Eingegangen	Deutschland	
1	Bestellung versendet	Versendet	Deutschland	Hermnes
2	Bestellung eingegangen	Eingegangen	Deutschland	
2	Lagerbestand prüfen	Eingegangen	Deutschland	
2	Artikel anfordern	Warte auf Artikel	China	COSCO Shipping
2	Artikel anfordern	Warte auf Artikel	Singapur	Neptun Orient Lines
2	Artikel anfordern	Warte auf Artikel	China	Hapag-Lloyd
2	Artikel eingegangen	Versendet	Deutschland	DHL
...	...	...	...	...

Tabelle 1: Beispiel Eventlog einer Bestellung

Gegeben ist der Eventlog aus Tabelle 1, der einen Bestellprozess eines deutschen Versandhandels darstellt. Neben der Fall- und Aktivitäts-Id enthält der Eventlog auch Informationen wie den Bestellstatus, der dem Kunden angezeigt wird, das Land, in dem sich der bestellte Artikel befindet, und das Speditionsunternehmen, das den Artikel transportiert.

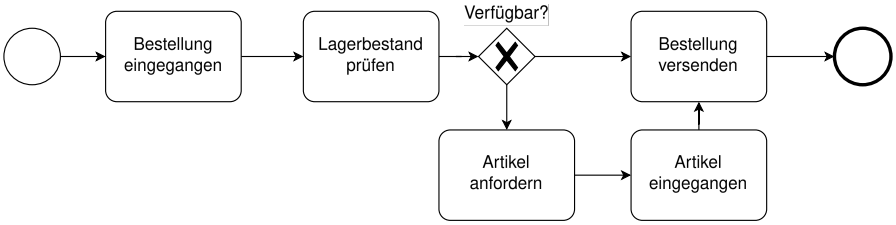


Abbildung 1: Prozess-Discovery des Eventlogs unter Verwendung der Fall- und Aktivitäts-Id

Bei der Durchführung von Process Discovery auf diesem Eventlog unter Verwendung der korrekten Fall- und Aktivitäts-Id wird der zugrunde liegende Prozess sichtbar. Dieser wird in Abbildung 1 dargestellt und zeigt den Ablauf der Aktionen im Prozess. Eine Bestellung geht in dem System ein. Ist die Ware im Lager verfügbar, wird diese direkt an den Kunden versendet. Andernfalls muss sie angefordert werden. Nachdem die Ware geliefert wurde, kann sie an den Kunden versendet werden.

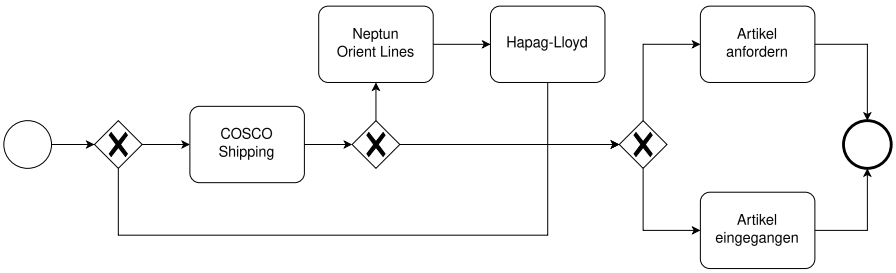


Abbildung 2: Prozess-Discovery des Eventlogs unter Verwendung des Spediteurs als Aktivitäts-Id

Die automatisierte Identifikation von Fall- und Aktivitäts-Id könnte ergeben, dass neben der eigentlichen Aktivitäts-Id die Felder *Spediteur* und *Land* Charakteristiken einer Aktivitäts-Id aufweisen. Im Eventlog aus Tabelle 1 wird bei der Aktivität *Artikel anfordern* ein neues Ereignis erzeugt, sobald der bestellte Artikel das Transportmittel bzw. den Spediteur wechselt. Wenn der Spediteur als Aktivitäts-Id für Prozess-Discovery verwendet wird, könnte ein Prozess entstehen, der Abbildung 2 ähnelt. Ist der Artikel im Lager vorhanden, wird dieser direkt mit einem deutschen Spediteur versendet. Ist dies nicht der Fall, muss dieser zunächst aus dem Ausland bezogen werden. Je nach Ursprungsland sind hierfür ein oder mehrere Speditionsunternehmen vonnöten. Dieser Prozess ermöglicht es, potentielle Probleme und Engpässe in der Lieferkette aufzudecken. Analysen können zum Beispiel zeigen, dass Verzögerungen gehäuft bei der Zusammenarbeit zweier bestimmter Spediteure auftreten.

Neben dem Spediteur lässt sich auch das Land als Aktivitäts-Id verwenden. Das Ergebnis ist ein Prozess, der zeigt, welche Länder die Artikel bis zur Lieferung an den Kunden durchlaufen. Auch hier ist es möglich, Engpässe zu identifizieren und beispielsweise ein Land oder die Interaktion mehrerer Länder in der Lieferkette zukünftig zu vermeiden.

Beide Beispiele zeigen eine neue Perspektive auf denselben Prozess, aus der Informationen gewonnen werden können, die zuvor nicht sichtbar waren. Dies schafft einen tieferen Einblick in die Struktur eines Prozesses und ermöglicht die Identifikation neuer Potenziale für Optimierungen.

Zielsetzung Eine automatische Identifikation von Fall- und Aktivitäts-Id hat somit eine reale Praxisrelevanz, da diese sowohl zu einer Kosten- und Ressourcenreduktion führt, als auch neue Potenziale der Prozessoptimierung eröffnet. Das Ziel dieser Arbeit ist die Erarbeitung eines Verfahrens, dass diese automatisierte Identifikation ermöglicht. Es soll folgende Forschungsfrage beantwortet werden:

Wie können Felder für Fall- und Aktivitäts-Id in unlabeled Eventlogs automatisiert identifiziert werden?

1.2 Vorgehen

Zunächst werden in Abschnitt 2 die Grundlagen von Process-Mining und die Herausforderungen von unlabeled Eventlogs beschrieben. Zudem wird erörtert, wie sich der in dieser Arbeit vorgestellte Ansatz in bestehende Arbeiten eingliedert und von bestehenden Lösungen abgrenzt.

In Abschnitt 3 werden die Grundlagen des entwickelten Konzepts zur Identifikation der Fall- und Aktivitäts-Id vorgestellt. Nach diesem theoretischen Kapitel folgt die konkrete Umsetzung und Implementierung des Konzepts in Form eines Prototyps.

Hierfür wird in Abschnitt 4 beschrieben, wie ein Eventlog eingelesen und vorverarbeitet werden kann.

Das Konzept verwendet maschinelles Lernen zur Identifikation der Fall- und Aktivitäts-Id. Da ein Mangel an Trainingsdaten besteht, wird in Abschnitt 5 ein System vorgestellt, das synthetische Eventlogs erzeugt, die als Datenbasis für das Anlernen der Modelle dienen.

In Abschnitt 6 wird das intelligente System beschrieben. Mehrere Strategien werden vorgestellt, die sich im Vorgehen zur Prognose der Fall- und Aktivitäts-Id unterscheiden. Das Kapitel schließt mit einer detaillierten Evaluation dieser Strategien unter Verwendung unterschiedlicher Datensätze ab.

Das vorgestellte Konzept basiert auf Annahmen, die mit der Implementierung überprüft werden. In Abschnitt 7 wird untersucht, ob diese Annahmen belegt werden.

Zuletzt werden in Abschnitt 8 die Ergebnisse dieser Arbeit zusammengefasst und weitere potenzielle Untersuchungen und Erweiterungen des Konzepts vorgestellt.

2 Grundlagen

2.1 Geschäftsprozesse

Geschäftsprozesse dienen der Standardisierung von Aktionsreihenfolgen. Sie ermöglichen es, komplexe Abläufe in klar definierte Aktivitäten zu unterteilen, um Effizienz, Qualität und Leistung zu verbessern. Oftmals werden Business Process Management Systeme (BPMS) eingesetzt, um Prozesse zu modellieren und durchzuführen. Ein einzelner Durchlauf eines Prozesses wird Prozessinstanz genannt. Wird beispielsweise ein Bestellprozess in einem BPMS modelliert, so entspricht eine Prozessinstanz einer einzelnen Bestellung. Jede eingehende Bestellung sorgt für die Erzeugung einer neuen Prozessinstanz. Im System können mehrere Prozessinstanzen parallel zueinander existieren, die alle unterschiedlich weit fortgeschritten sind [24].

Geschäftsprozesse enthalten unterschiedliche Elemente, die den Prozessfluss definieren oder die getätigten Aktionen beschreiben. Im folgenden sind die für diese Arbeit wichtigsten Elemente aufgezählt:

Start- und Endknoten definieren den Beginn bzw. das Ende eines Prozesses. Jede Prozessinstanz beginnt an dem Startknoten. Wird der Endknoten erreicht, gilt die Prozessinstanz als beendet. Es kann jeweils nur einen Start- und einen Endknoten in einem Geschäftsprozess existieren [24].

Aktivitäten sind die grundlegenden Aktionen oder Handlungen, die innerhalb eines Geschäftsprozesses durchgeführt werden. Aktivitäten können manuell oder automatisiert ausgeführt werden und erfordern oft den Einsatz von Ressourcen wie Arbeitskräften, Maschinen oder Informationstechnologien [24].

Entscheidungen beeinflussen den weiteren Verlauf eines Prozesses maßgeblich. Sie ermöglichen das Durchlaufen unterschiedlicher Abschnitte basierend auf festgelegten Bedingungen. Entscheidungen können beispielsweise durch Ereignisse, Analysen oder Systemzustände beeinflusst werden und sowohl automatisiert als auch manuell erfolgen [24].

Parallele Gateways ermöglichen die simultane Ausführung mehrerer Aktivitäten. Der Prozessverlauf wird in mehrere Pfade aufgeteilt, die parallel zueinander bearbeitet werden. Zu einem späteren Zeitpunkt existiert ein zweites Paralleles Gateway, welches die Pfade wieder vereinigt. An diesem wird Prozessausführung pausiert, bis alle nebenläufigen Pfade abgeschlossen sind [24].

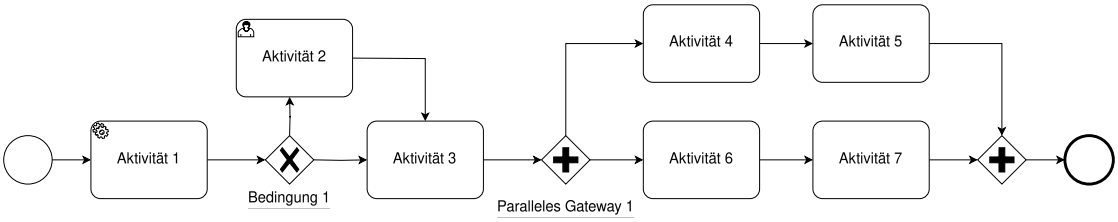


Abbildung 3: Beispiel eines Prozesses in der BPMN Notation

In Abbildung 3 wird beispielhaft ein Prozess als BPMN (*Business Process Model and Notation*) dargestellt, der alle 4 beschriebenen Elemente enthält. Start- und Endknoten werden als Kreise dargestellt. Der Startknoten hat immer einen dünneren Rand als der Endknoten. Aktivitäten werden als Rechtecke mit abgerundeten Ecken dargestellt. Sie können im oberen linken Bereich Symbole enthalten, die auf eine manuelle oder automatisierte Ausführung hinweisen. Entscheidungen werden als Rauten dargestellt, welche ein *X* im Zentrum enthalten. In der Darstellung ist die *Aktivität 2* optional, da dessen Ausführung von *Entscheidung 1* abhängig ist. Parallele Gateways werden durch eine Raute mit einem Plus in der Mitte dargestellt. Zu beachten ist, dass nur die Pfade, die *Paralleles Gateway 1* verlassen, parallel zueinander ausgeführt werden, nicht die Elemente dieser Pfade. Dies bedeutet, dass *Aktivität 4* parallel zu *Aktivität 6* ausgeführt wird, allerdings *Aktivität 5* erst beginnt, wenn *Aktivität 4* beendet ist.

2.2 Eventlogs

Wenn die Abläufe in einem System durch standardisierte Prozesse strukturiert sind, werden die durchgeführten Aktionen und Ereignisse in der Regel in Eventlogs festgehalten. Dies ermöglicht eine Transparenz und Nachvollziehbarkeit von Problemen, da vergangene Prozessinstanzen protokolliert und einsehbar sind. Zusätzlich dienen Eventlogs als Datengrundlage für Prozess-Mining-Verfahren. Prozess-Mining beschäftigt sich mit der Analyse von Eventlogs, um neue Potenziale und Optimierungen in den Prozessen zu identifizieren [8, 18]. Beispielsweise kann der Prozess anhand des Eventlogs rekonstruiert

(*Process-Discovery*), der geplante mit dem tatsächlichen durchgeführten Prozess verglichen (*Conformance-Checking*) oder Engpässe und Prozessoptimierungen erkannt werden (*Process Enhancement*).

Ein Eventlog setzt sich aus einer Reihe von Ereignissen zusammen. Jede ausgeführte Aktivität in einem Geschäftsprozess führt zu der Erzeugung eines Ereignisses, welches üblicherweise aus mindestens 3 Feldern besteht [8]:

- Die Aktivitäts-Id, die eine spezifische Aktivität im Geschäftsprozess eindeutig identifiziert.
- Die Fall-Id, die eine eindeutige Zuordnung zu einer ausgeführten Prozessinstanz ermöglicht.
- Der Zeitpunkt der Ausführung der Aktivität. Eventlogs können auch zwei Zeitstempel enthalten, die den Start und das Ende einer Aktivität markieren.

Neben diesen grundlegenden Feldern können Ereignisse zusätzliche Informationen enthalten, die weitere Einblicke in die durchgeführten Aktivitäten oder den Zustand des zugrundeliegenden Systems bieten. In dieser Arbeit werden diese ergänzenden Informationen als Attribute bezeichnet. Diese Attribute können in verschiedenen Datentypen vorliegen, wie etwa Text, Zeitstempel, Zahlen oder Binärdaten und sowohl automatisch als auch manuell erfasst werden. Beispiele für typische Attribute sind Fehlercodes, Kundennummern oder der Name des Anwenders, der die Aktivität ausgeführt hat. Fall- und Aktivitäts-Id lassen sich als spezielle Attribute betrachten, die strukturelle Informationen des Geschäftsprozesses enthalten [8, 18].

2.2.1 Heterogene Eventlogs

Von heterogenen Eventlogs wird gesprochen, wenn in einem System mehrere getrennte Eventlogs aus verschiedenen Applikationen existieren. In einem Unternehmen können bspw. in verschiedenen Abteilungen unterschiedliche BPMS zum Einsatz kommen. Das Resultat sind multiple Eventlogs, welche sich in Semantik, Form und Syntax unterscheiden [26]. In diesen Fällen kann nicht davon ausgegangen werden, dass dieselben eindeutigen Ids systemweit in jedem Eventlog existieren. Die Herausforderung von heterogenen Eventlogs besteht darin, eine Verknüpfung zwischen den einzelnen Eventlogs herzustellen, um diese zu einem übergeordneten Eventlog zusammenzufassen, der alle Inhalte kombiniert. Dies ist keine triviale Aufgabe, da in realen Daten oftmals nur wenige Einträge

auf Verbindungen zwischen den Eventlogs hinweisen. Auch können unterschiedliche Zeichenkodierungen, Homonyme und Synonyme oder sprachabhängige Werte die Zuordnung erschweren [26].

2.2.2 Unlabeled Eventlogs

Für die Durchführung von Prozess-Mining werden strukturierte (engl. *labeled*) Eventlogs benötigt, die die vollständigen Informationen zu Fall-Id, Aktivitäts-Id sowie einen Zeitstempel enthalten [18, 8]. In der Praxis kann es jedoch vorkommen, dass diese Informationen fehlerhaft, unvollständig oder gänzlich abwesend sind. Dies kann oftmals auf eine manuelle Datenerfassung oder eine heterogene Systemlandschaft ohne zentrales Loggingsystem zurückgeführt werden [18]. Diese Eventlogs werden als *unlabeled* bezeichnet und lassen sich nicht mit herkömmlichen Prozess-Mining-Verfahren verarbeiten [18].

Der Themenkomplex der unlabeled Eventlogs lässt sich in zwei Problemstellungen untergliedern:

Fehlende Felder Diese betreffen Eventlogs, in denen die Felder für Aktivitäts-Id, Fall-Id oder Zeitstempel gänzlich fehlen oder teilweise unvollständig sind. Verfahren, die sich mit diesen Eventlogs befassen, lassen sich in 2 Kategorien einteilen [21, 15, 6].

- Es wird versucht die unlabeled Eventlogs zu labeled Eventlogs umzuwandeln. Oftmals werden Zusatzinformationen, wie beispielsweise die Ausführungsdauer der Aktivitäten oder der Aufbau des Geschäftsprozesses, verwendet, um die fehlenden Felder zu rekonstruieren. Die resultierenden labeled Eventlogs lassen sich daraufhin mit herkömmlichen Prozess-Mining-Verfahren verarbeiten.
- Die Entwicklung neuer Prozess-Mining-Verfahrenen, welche nicht auf die fehlenden Informationen angewiesen sind. Ein Beispiel hierfür ist der *correlation miner*, der auf Eventlogs ohne Fall-Id angewendet werde kann [18].

Unbekannte Felder Es kann vorkommen, dass Eventlogs die Informationen für Aktivitäts-Id, Fall-Id und Zeitstempel zwar enthalten, allerdings unbekannt ist, in welchen Attributen diese stehen. Dies tritt insbesondere bei der Arbeit mit heterogenen Eventlogs auf, bei der die Konsistenz der Datenfelder über verschiedene Quellen hinweg nicht gewährleistet werden kann [26]. Das Attribut, dass die Aktivitäts-Id eines Eventlogs

enthält, könnte in einem zweiten Eventlog eine andere Bedeutung besitzen. Ziel dieser Verfahren ist die korrekte Identifikation der Attribute, die Fall-Id, Aktivitäts-Id oder den Zeitstempel repräsentieren [7]. Sind diese Erkannt, können herkömmliche Prozess-Mining-Verfahren angewendet werden. Die Aufgabenstellung dieser Arbeit gliedert sich in diesen Problemkomplex ein.

2.3 Maschinelles Lernen

Maschinelles Lernen ist ein Teilgebiet der künstlichen Intelligenz, das sich mit der Entwicklung von Algorithmen und statistischen Modellen befasst, die Zusammenhänge und Muster aus Daten automatisiert erlernen können. Dies ermöglicht es den Modellen, anhand neuer, unbekannter Daten Entscheidungen zu treffen, Schlussfolgerungen zu ziehen oder Prognosen für zukünftige Ereignisse abzugeben, ohne explizit für diese Aufgabenstellung programmiert zu sein [4].

Maschinelles Lernen kann in 3 Bereiche unterteilt werden: das überwachte Lernen (*supervised learning*), das unüberwachte Lernen (*unsupervised learning*) und das bestärkende Lernen (*reinforcement learning*) [4].

2.3.1 Überwachtes Lernen

Beim überwachten Lernen wird ein Modell anhand eines Trainingsdatensatzes trainiert, bei dem das erwartete Ergebnis bekannt ist. Ziel ist es, eine Funktion zu erlernen, die die Eingabedaten mit den erwarteten Ergebnissen in Beziehung setzt, sodass das Modell in der Lage ist, korrekte Vorhersagen für Daten mit unbekanntem Ergebnis zu treffen [4]. Typische Technologien, die für das überwachte Lernen eingesetzt werden, sind lineare Regression, Support Vector Machines, neuronale Netze und Entscheidungsbäume [4].

Overfitting Overfitting tritt auf, wenn ein Modell zu gut an die Trainingsdaten angepasst ist und dadurch seine Fähigkeit verliert, auf neue, ungesehenen Daten zu interpretieren. Ein überangepasstes Modell hat die zugrunde liegenden Muster in den Trainingsdaten „auswendig gelernt“ und passt sich an das Rauschen oder die irrelevanten Details der Trainingsdaten an. Overfitting tritt besonders häufig auf, wenn eine ungenügende Anzahl an Trainingsdaten verwendet wird oder sehr komplexe Modelle auf einfache Aufgabenstellungen angewendet werden [4].

Voreingenommenheit der Stichproben Eine Voreingenommenheit der Stichproben (*sampling bias*) tritt auf, wenn die Trainingsdaten nicht repräsentativ für die tatsächlichen Einsatzbedingungen des Modells sind. Ein klassisches Beispiel ist die Bildklassifikation, bei der prognostiziert werden soll, ob ein Hund oder eine Katze in einem Bild dargestellt wird. Wenn alle Bilder von Hunden im Freien aufgenommen werden, während alle Katzenbilder in Innenräumen entstehen, lernt das Modell möglicherweise, den Hintergrund (drinnen vs. draußen) als Unterscheidungsmerkmal zu verwenden, anstatt die eigentlichen Merkmale von Hunden und Katzen. Dies führt zu einem Modell, das gut auf den Trainingsdaten funktioniert, aber schlechte Ergebnisse liefert, wenn es mit neuen, andersartigen Daten konfrontiert wird [4].

2.3.2 Entscheidungsbäume

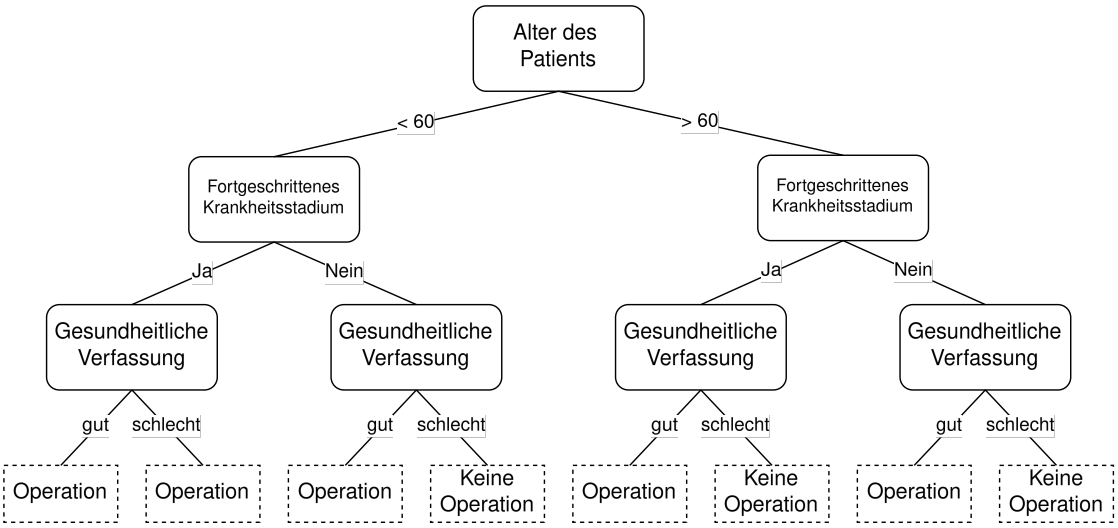


Abbildung 4: Beispiel eines Prozesses in der BPMN Notation

Ein Entscheidungsbaum ist ein Modell, das eine hierarchische, baumähnliche Knotenstruktur aufweist. Jeder Knoten im Baum stellt eine Entscheidung dar, die auf einem bestimmten Merkmal der Eingangsdaten basiert. Durch die Zweige werden die Ergebnisse dieser Entscheidungen dargestellt. Die Blätter des Baums repräsentieren die Prognose des Entscheidungsbaums [4].

In Abbildung 4 wird ein Beispiel für einen Entscheidungsbaum dargestellt. Es wird prognostiziert, ob ein Patient operiert werden soll, anhand der Merkmale des Alters, des Fortschritts der Krankheit und der generellen gesundheitlichen Verfassung. Die Knoten

werden durch Rechtecke mit abgerundeten Kanten dargestellt, die Zweige durch Linien zwischen den Knoten und die Blätter durch Rechtecke mit gestricheltem Rand. Für einen Patienten, der älter als 60 Jahre ist, sich im fortgeschrittenen Krankheitsstadium befindet und eine schlechte körperliche Verfassung aufweist, prognostiziert der Entscheidungsbaum, dass nicht operiert werden sollte.

Die Hauptvorteile von Entscheidungsbäumen sind ihre Einfachheit und Interpretierbarkeit. Sie sind leicht zu visualisieren und zu verstehen, was sie zu einem nützlichen Werkzeug für die explorative Datenanalyse macht. Allerdings neigen Entscheidungsbäume dazu, sehr tief und komplex zu werden, was zu Overfitting führen kann [10].

2.3.3 Random Forests

Random Forests sind ein populärer Ensemble-Lernalgorithmus, der aus einer Vielzahl von Entscheidungsbäumen besteht. Jeder Entscheidungsbaum im Wald wird auf einer zufällig gewählten Teilmenge des Trainingsdatensatzes trainiert. Bei der Vorhersage wird das Durchschnittsergebnis (bei Regression) oder die Mehrheit der klassifizierten Ergebnisse der Bäume verwendet [10].

Die Haupteigenschaften von Random Forests sind:

- **Robustheit:** Random Forests sind weniger anfällig für Overfitting im Vergleich zu einzelnen Entscheidungsbäumen, da mehrere Entscheidungsbäume auf Teilmengen des Datensatzes trainiert werden, wodurch die Varianz reduziert wird [10].
- **Feature-Importanz:** Random Forests bieten eine Methode zur Schätzung der Wichtigkeit von Merkmalen, die dabei hilft, die wichtigsten Prädiktoren in einem Datensatz zu identifizieren [10].
- **Vielseitigkeit:** Sie können sowohl für Klassifikations- als auch für Regressionsprobleme verwendet werden und sind in der Lage, mit großen Datensätzen und einer großen Anzahl von Features umzugehen [10].

2.4 Stand der Technik

In diesem Kapitel werden bestehende Arbeiten vorgestellt, die sich mit der Thematik der unlabeled Eventlogs beschäftigen.

Fehlende Fall-Id Die Arbeit [8] untersucht unlabeled Eventlogs mit bekannter Aktivitäts-Id und fehlender Fall-Id. Hierbei sind der zugrundeliegende Prozess und Heuristiken, wie die minimale, maximale und durchschnittliche Ausführungszeit der Aktivitäten bekannt. Das Verfahren verwendet Entscheidungsbäume, um aus diesen Informationen eine künstliche Fall-Id zu generieren.

In [18] wird der *correlation miner* vorgestellt, welcher Process Discovery ohne Fall-Id ermöglicht. Die Zuordnung geschieht über den Zeitstempel. Tritt in einem Eventlog immer eine Aktivität mit konstantem Abstand hinter einer anderen auf, kann davon ausgegangen werden, dass die Aktivitäten im Geschäftsprozess aufeinander folgen. Über diese Abhängigkeiten kann der gesamten Geschäftsprozess rekonstruieren werden.

In der Arbeit [21] wird ein ähnlichen Ansatz verwendet, allerdings steht nicht Process Discovery im Mittelpunkt, sondern die Generierung einer künstlichen Fall-Id. Ein Eventlog wird auf wiederkehrende Muster untersucht. Alle Ereignisse eines Musters können als Prozessinstanz angesehen werden und erhalten dieselbe künstliche Fall-Id.

Unbekannte Aktivitäts-Id Oftmals ist die Granularität der Eventlogs feiner als die der Aktivitäten des Geschäftsprozesses. Zu einer Aktivität können mehrere Einträge im Eventlogs existieren. Ist die Aktivitäts-Id nicht bekannt, können die Aktivitäten nicht den Ereignissen zugeordnet werden. Es wird vom sogenannten Granularitätsproblem gesprochen. In [6] entwickeln Forscher ein Verfahren, dass eine Zuordnung der Einträge des Eventlogs zu den Aktivitäten auch ohne Aktivitäts-Id ermöglicht.

In [26] werden heterogene Eventlogs untersucht mit dem Ziel eine Zuordnung zwischen diesen herzustellen. In Unternehmen kann es vorkommen, dass derselbe Geschäftsprozess in unterschiedlichen Abteilungen mit verschiedenen BPMS modelliert wird. In diesem Kontext existiert ebenfalls das Granularitätsproblem, da nicht davon ausgegangen werden kann, dass beide Geschäftsprozesse mit dem selben Detailgrad modelliert sind.

Unbekannte Fall- und Aktivitäts-Id Die Problemstellung bei der sowohl Fall-Id als auch Aktivitäts-Id existieren, allerdings nicht bekannt ist, welche Felder im Eventlog diese enthalten, ist nur unzureichend erforscht. Einzig die Arbeit [7] befasst sich mit dieser Aufgabenstellung. Eventlog weisen besondere Charakteristiken in der Wertwiederholung der Fall- und Aktivitäts-Id auf. Eine Prozessinstanz erzeugt viele Eventlog-Einträge mit gleicher Fall-Id und Aktivitäten werden von vielen Prozessinstanzen durchlaufen. Daher

enthalten beide Felder einzeln betrachtet eine hohe Repetition, d.h. die Spalte der Fall-Id und Aktivitäts-Id enthalten viele gleiche Werte [7].

Die Kombination beider Spalten weist allerdings nur eine geringe Wertwiederholung auf, da eine Aktivität in einer Prozessinstanz in der Regel nur einmal durchlaufen wird. Diese charakteristischen Merkmal werden von den forschenden genutzt, um die korrekte Fall- und Aktivitäts-Id zu identifizieren [7].

In dieser Arbeit wird dieselbe Problemstellung wie in [7] behandelt. Das in [7] vorgestellte Konzept ist allerdings auf den Anwendungsfall einer spezifischen *Middleware*-Anwendung ausgelegt und wird nur auf einem einzelnen Eventlog aus diesem System validiert. Die Identifikation der Fall- und Aktivitäts-Id erfolgt ausschließlich über die Charakteristiken der Wertwiederholungen dieser Ids. Es wird nicht darauf eingegangen, dass Attribute in den Eventlogs ebenfalls diese charakteristischen Wertwiederholungen aufweisen können. Daher wird in dieser Arbeit ein eigener Ansatz vorgestellt, der neben der Wertwiederholung weitere Merkmale zur Identifikation der Fall- und Aktivitäts-Id verwendet. Dies soll dazu führen, dass der Ansatz in unterschiedlichen Einsatzgebieten zu robusteren Ergebnissen führt.

3 Identifikation von Fall- und Aktivitäts-Id

Dieses Kapitel erläutert das Konzept, mit dem die Identifikation der Fall- und Aktivitäts-Id in unlabeled Eventlogs ermöglicht wird. Es werden die theoretischen Grundlagen des Verfahrens beschrieben, wie Muster und Sequenzen in Eventlogs entstehen und wie diese genutzt werden können, um die Fall- und Aktivitäts-Id zu identifizieren.

3.1 Muster und Sequenzen

Die Identifikation von Fall- und Aktivitäts-Id dieser Arbeit stützt sich auf der Beobachtung, dass Fall- und Aktivitäts-Ids wiederkehrende Muster in Eventlogs erzeugen [21]. Zur Analyse dieses Phänomens werden die Begriffe *Sequenz* und *Muster* definiert.

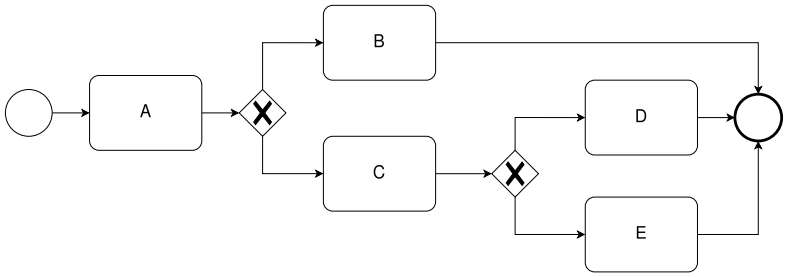


Abbildung 5: Beispielprozess mit den Pfaden *A-B*, *A-C-D* und *A-C-E*

Sequenzen Durch einen Geschäftsprozess führt nur eine endliche Anzahl an Pfaden. Ein Pfad beschreibt eine einzigartige Abfolge an Aktivitäten, die vom Start- zum Endknoten führen. In Abbildung 5 wird ein Geschäftsprozess dargestellt, der die Pfade *A-B*, *A-C-D* und *A-C-E* enthält.

Wird ein Prozess entlang eines Pfades ausgeführt, so entsteht für jede Aktivität ein Eintrag im Eventlog. Diese enthalten eine Fall-Id, die für alle Einträge derselben Prozessinstanzen identisch ist, den Namen der ausgeführten Aktivität als Aktivitäts-Id und einen Zeitstempel [8]. Werden die Einträge einer Prozessinstanz chronologisch nach dem Zeitstempel sortiert, so spiegelt die Spalte der Aktivitäts-Id die Aktivitätsreihenfolge des durchlaufenen Pfades wider.

In dieser Arbeit wird der Begriff Sequenz als ein Tuple der Form $(cId, [aId_1, aId_2, ..., aId_n])$ definiert, wobei cId der Fall-Id und $[aId_1, aId_2, ..., aId_n]$ der chronologischen Aktivitätsreihenfolge der Prozessinstanzen entspricht. In Tabelle 2 wird ein Auszug des Eventlogs

für den in Abbildung 5 dargestellten Prozesses, aufgelistet. Dieser enthält die Sequenzen $(0, [A, B]), (1, [A, C, D])$ und $(2, [A, B])$.

Fall-Id	Activity-Id	Zeitstempel
0	A	00:00
0	B	00:01
1	A	00:00
1	C	00:01
1	D	00:02
2	A	00:03
2	B	00:04

Tabelle 2: Ein möglicher Eventlog für den Prozess aus Abbildung 5

Muster Jede Prozessinstanz generiert eine einzigartige Sequenz im Eventlog. Wenn ein Prozess mehrfach entlang desselben Pfades ausgeführt wird, resultiert dies in Sequenzen mit identischer Aktivitätsreihenfolge. Dies wird in Tabelle 2 am Beispiel des Pfades *A*-*B* deutlich, der für die Fall-Id 0 und die Fall-Id 2 durchlaufen wird. In dieser Arbeit wird der Begriff Muster verwendet, um eine Aktivitätsreihenfolge zu beschreiben, die mehrfach im Eventlog vorkommt. Ein Muster wird als ein Tupel der Form $([aId_1, aId_2, \ldots, aId_m], [cId_1, cId_2, \ldots, cId_n])$ definiert. Es enthält die Aktivitätsreihenfolge $[aId_1, aId_2, ..., aId_m]$ und die Fall-Ids $[cId_1, cId_2, ..., cId_n]$, die diese Aktivitätsreihenfolge verursacht haben.

3.2 Charakteristiken von Geschäftsprozessen

Im vorherigen Unterabschnitt wird beschrieben, wie durch die Fall- und Aktivitäts-Id Muster in den Eventlogs entstehen. Diese Muster enthalten sowohl strukturelle Informationen über dem zugrunde liegenden Prozess, als auch Informationen über die Prozessausführung, da sowohl die Pfade (Aktivitätsreihenfolgen) als auch die Anzahl der Durchläufe jedes Pfades bekannt sind.

Neben der Fall- und Aktivitäts-Id können auch zwei beliebige Attribute des Eventlogs verwendet werden, um Sequenzen und Muster zu erkennen. Ein der Attribut wird dabei wie eine Fall-Id behandel und das andere wie eine Aktivitäts-Id. Diese Muster enthalten ebenfalls strukturelle Informationen zu einem Prozess, der durch diese Attribute aufgespannt wird. Da sich die Wertwiederholungen der Attribute von denen der Fall- und

Aktivitäts-Id unterscheiden, weicht der durch diese Attribute dargestellte Prozess von dem eigentlichen Geschäftsprozess ab.

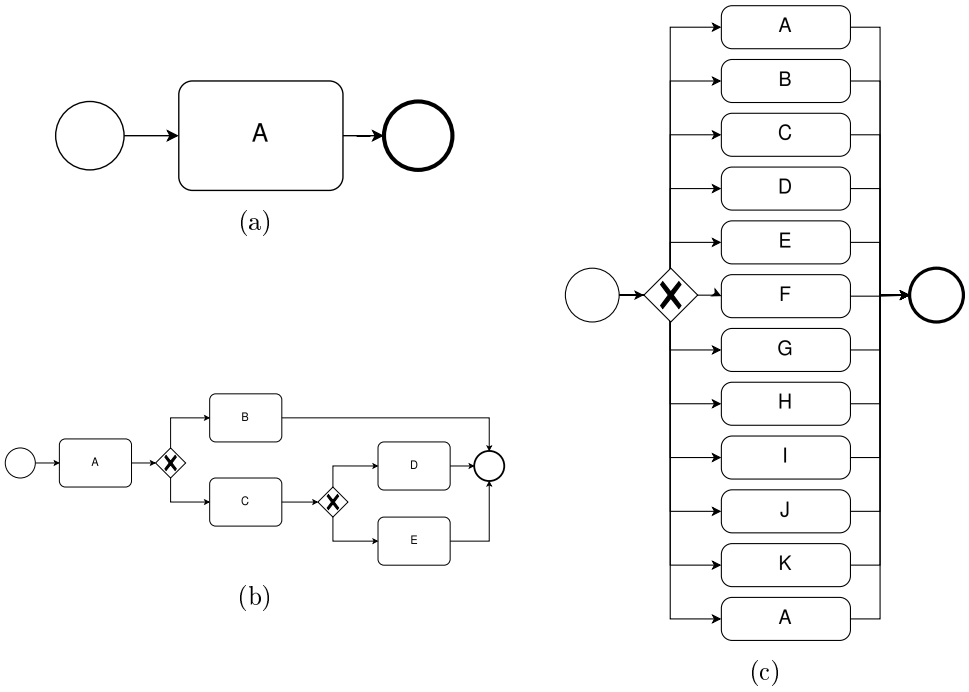


Abbildung 6: Beispiele für Muster in Eventlogs unter Verwendung verschiedener Felder

Dies wird anhand eines Beispiels verdeutlicht. Aus einem Eventlog werden 2 Paare bestehend aus 2 Attributen und ein Paar bestehend aus der korrekten Fall- und Aktivitäts-Id ausgewählt. Für jedes Paar wird der Eventlog auf Muster untersucht und die strukturellen Informationen der Muster als BPMN dargestellt.

In Abbildung 6 werden die 3 resultierenden Prozesse abgebildet. Es wird die Frage gestellt, welcher der Prozesse aus den Mustern der Fall- und Aktivitäts-Id generiert wurde. Die Antwort dieser Frage ist offensichtlich und kann mit Abbildung 6b beantwortet werden. Dieser Prozess weißt die größte Ähnlichkeit zu einem Geschäftsprozess auf, da dieser strukturelle Charakteristiken enthält, die einem Geschäftsprozess ähneln. Zusätzlich weisen die beiden anderen Prozesse Strukturen auf, die untypisch für einen Geschäftsprozess sind, da Abbildung 6a nur eine Aktivität enthält und Abbildung 6c aus nur einer große Bedingung besteht.

Das Konzept dieser Arbeit zur Identifikation der Fall- und Aktivitäts-Id basiert im Kern auf dieser Vorgehensweise. Es wird angenommen, dass Geschäftsprozesse besondere Cha-

rakteristiken aufweisen, da diese bestimmten Regeln und Normen unterliegen, die sowohl die Prozessstruktur als auch die Prozessdurchführung beeinflussen. Diese Charakteristiken spiegeln sich auch in den Mustern wider, da diese die strukturellen Informationen zum Prozessaufbau und der Prozessdurchführung enthalten. Es wird zudem angenommen, dass diese Charakteristiken identifiziert und analysiert werden können. Dies ermöglicht es, die Muster auf diese Charakteristiken zu untersuchen und somit zu bestimmen, ob die Muster durch die Fall- und Aktivitäts-Id ausgelöst werden. Neben den Charakteristiken, die ein Indiz für die korrekte Fall und Aktivitäts-Id sind, können auch Charakteristiken existieren, die ausschließen, dass es sich bei den verwendeten Eventlogfeldern um Fall- und Aktivitäts-Id handelt.

Analyse eines unlabeled Eventlogs Um mit diesem Konzept in einem Eventlog die unbekannte Fall- und Aktivitäts-Id zu identifizieren, werden zunächst alle möglichen Paarkombinationen aus den Attributen des Eventlogs gebildet. Unter diesen Paaren befindet sich genau eines, das die korrekten Fall- und Aktivitäts-Id enthält. Für jedes Paar wird der Eventlog auf Sequenzen und Muster untersucht. Diese Muster werden auf Charakteristiken analysiert, die für oder gegen einen Geschäftsprozess sprechen. Das Paar, dessen Muster einem Geschäftsprozess am meisten ähneln, wird als Fall- und Aktivitäts-Id angesehen.

3.2.1 Analyse der Muster

Damit die Muster auf die Charakteristiken der Geschäftsprozesse untersucht werden können, müssen diese quantifiziert werden. Dies geschieht durch die Berechnung von *Kennzahlen*, die die strukturellen Informationen des Prozessaufbaus und der Prozessdurchführung in gebündelter Form repräsentieren. Beispiele für solche Kennzahlen sind die Länge der Muster, ihre Auftrittshäufigkeit oder die Ähnlichkeit der Aktivitätsreihenfolgen untereinander. Alle Kennzahlen sowie deren Berechnung werden in Unterunterabschnitt 4.3.4 beschrieben.

3.2.2 Analyse der Charakteristiken

Geschäftsprozesse sind komplexe Konstrukte. Die im vorherigen Abschnitt beschriebenen Charakteristiken, die Geschäftsprozesse aufweisen, sind daher auch nicht an statische Regeln gebunden. Ein großer und komplexer Geschäftsprozess weist andere Charakteristiken

auf als ein kleiner und einfacher. Die aus den Mustern errechneten Kennzahlen können sich daher erheblich, basierend auf dem zugrundeliegenden Prozess, unterscheiden.

Ein Beispiel hierfür ist die Anzahl der Pfade. In einem kleinen und einfachen Prozess wird erwartet, dass dieser Wert gering ist. Große und komplexe Prozesse weisen dagegen viele Pfade auf.

Dieses Verhalten verhindert, dass ein einfaches regelbasiertes System für die Auswertung der Kennzahlen verwendet werden kann. Erst durch das Betrachten der Verhältnisse und Beziehungen der Kennzahlen untereinander können Indizien gesammelt werden, die dafür oder dagegen sprechen, dass die Muster durch die Fall- und Aktivitäts-Id ausgelöst werden.

Diese Abhängigkeiten manuell zu erforschen, ist ein äußerst aufwändiger und komplexer Vorgang. Aus diesem Grund wird in dieser Arbeit maschinelles Lernen eingesetzt, um die Charakteristiken von Geschäftsprozessen automatisiert zu ermitteln. Die Interpretation der Kennzahlen kann als Klassifizierungsproblem betrachtet werden. Anhand der Kennzahlen wird klassifiziert, ob die zugrundeliegenden Muster aus einem Geschäftsprozess stammen.

In den meisten Eventlogs ist bekannt, welche Felder der Aktivitäts- und Fall-Id entsprechen. Gemäß Unterunterabschnitt 2.3.1 ermöglicht dies den Einsatz von überwachtem Lernen, da das erwartete Ergebnis im Voraus bekannt ist. Die Analyse der Kennzahlen mithilfe von maschinellem Lernen wird im Folgenden als *das intelligente System* bezeichnet.

3.3 Hindernisse der Musterbildung

Geschäftsprozesse und Eventlogs können Strukturen aufweisen, die die Bildung von Mustern beeinflusst. Diese sorgen dafür, dass die Komplexität und Variabilität eines Prozesses stark erhöht wird.

3.3.1 Rauschen

In der Praxis können Abweichungen zwischen dem Eventlog und dem tatsächlichen Geschäftsprozess auftreten [1]. Es wird vom sogenannte Rauschen gesprochen. Gründe für

Rauschen sind Vielseitig. Beispielsweise kann Rauschen auf eine manuelle Datenerfassung/-eingabe, technische Fehler oder abweichende Namenskonventionen zurückgeführt werden [11]. Es kann zudem vorkommen, dass Ausnahmen auftreten, die nicht durch den zugrundeliegenden Prozess abgedeckt sind. Die Folge sind Prozessinstanzen im Eventlog, die vom eigentlichen Prozessablauf abweichen [1].

Wenn bei der Durchführung einer Prozessinstanz ein ungültiges Ereignis festgehalten wird, Aktivitäten übersprungen, Aktionen vertauscht oder doppelt durchgeführt werden, führt dies zur Bildung von Sequenzen mit einer neuen Aktivitätsreihenfolge. Da Rauschen selten und sporadisch Auftritt, ist es unwahrscheinlich, dass dieselbe Aktivitätsreihenfolge erneut auftreten wird [1]. Somit werden durch Rauschen viele Muster erzeugt, die alle eine geringe Häufigkeit aufweisen.

Dies kann dazu führen, dass die errechneten Kennzahlen verfälscht werden, da diese auf den Mustern aufbauen. Aus diesem Grund werden Mechaniken der Rauschunterdrückung entwickelt und implementiert. Die Muster, die eine geringe Auftrittswahrscheinlichkeit aufweisen, werden erkannt und für die Errechnung der Kennzahlen ausgeschlossen.

3.3.2 Zyklen

Zyklen erlauben in Geschäftsprozessen die Wiederholung bereits durchlaufener Abschnitte [12]. Ein exemplarisches Minimalbeispiel eines Prozesses mit einer zyklischen Struktur wird in Abbildung 7 dargestellt. Theoretisch existieren in diesem Prozess unendlich viele Pfade vom Start- zum Zielknoten, da die Aktivität *A* beliebig oft wiederholt werden kann. Dies führt dazu, dass im Eventlog viele Sequenzen mit unterschiedlichen Aktivitätsreihenfolgen entstehen, die auf die variierende Anzahl der Durchläufe des Zyklus zurückzuführen sind. Eine solche Variabilität kann die Identifikation von Fall- und Aktivitäts-Id erschweren, da dies die Bildung von Mustern beeinträchtigt.

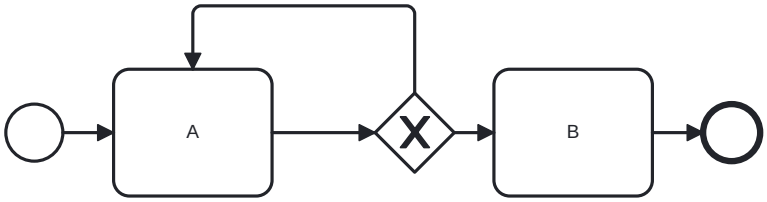


Abbildung 7: Minimalbeispiel eines Geschäftsprozesses mit Zyklus

Diese Problematik tritt allerdings nur auf, wenn die Verteilung der Zyklendurchläufe völlig willkürlich ist. In dieser Arbeit wird allerdings angenommen, dass eine solche zufällige Verteilung in der Praxis selten anzutreffen ist. Unternehmen streben danach, ihre Prozesse möglichst effizient zu gestalten. Ein Prozess, bei dem Zyklen häufig durchlaufen werden, weist eine längere Ausführungsdauer auf und verursacht somit höhere Kosten im Bezug auf Ressourcen wie Zeit, Geld und Personal. Daher ist zu erwarten, dass die Wahrscheinlichkeit für einen weiteren Zyklendurchlauf mit zunehmender Anzahl an Durchläufen sinkt.

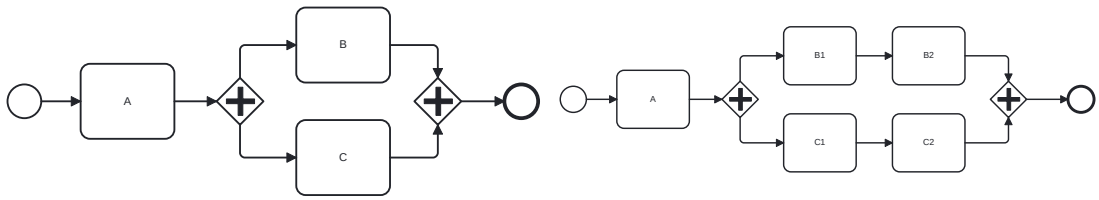
Zudem treten Zyklen häufig für Fehleridentifikation und Qualitätskontrolle auf. Eine Reihe von Aktivitäten wird durchgeführt und anschließend überprüft, ob Fehler aufgetreten sind oder Qualitätsanforderungen nicht erfüllt werden. Wird ein Mangel festgestellt, erfolgt die Wiederholung der Aktivitäten. Gerade in diesen Fällen kann angenommen werden, dass die Anzahl der Iterationen eines Zyklus nicht zufällig ist. Mängel treten typischerweise sporadisch auf, daher ist generell nur eine geringe Anzahl an Durchläufen des Zyklus zu erwarten. Nach der Korrektur eines Mangels verringert sich die Wahrscheinlichkeit weiterer Mängel bei der nächsten Überprüfung. Somit treten Zyklendurchläufe mit geringer Wiederholungszahl häufiger auf.

Sind in einem Eventlog viele Prozessinstanzen enthalten, führt dieses Verhalten dennoch zur Bildung von Mustern, da bestimmte Zyklendurchläufe überdurchschnittlich oft auftreten. Wird beispielsweise angenommen, dass der Zyklus aus Abbildung 7 typischerweise 2-3 Mal durchlaufen wird, entstehen Muster mit den Aktivitätsreihenfolgen $A - A - B$ und $A - A - A - B$ im Eventlog. Prozessinstanzen mit variierenden Zyklendurchläufen treten seltener auf und werden in dieser Arbeit als Rauschen betrachtet.

3.3.3 Nebenläufigkeit

Nebenläufigkeit in Geschäftsprozessen erlaubt die parallele Ausführung mehrerer Aktivitäten [12]. Dabei wird ein Prozesspfad in mehrere Subpfade unterteilt, die gleichzeitig bearbeitet werden. Ein Minimalbeispiel für einen Prozess mit Nebenläufigkeit wird in Abbildung 8a dargestellt. In diesem Beispiel erfolgt die gleichzeitige Ausführung der Aktivitäten B und C.

Nebenläufigkeit hat Auswirkungen auf die Sequenzen im Eventlog. Aufgrund variabler Ausführungszeiten von Aktivitäten in verschiedenen Prozessinstanzen kann die Reihenfolge der parallel ausgeführten Aktivitäten variieren [18]. Zum Beispiel könnte der in



(a) (b)

Abbildung 8: Minimalbeispiele für Prozesse mit Nebenläufigkeit

Abbildung 8a dargestellte Prozess die Aktivitätsreihenfolge $A - B - C$ generieren, falls die Ausführung von C länger dauert als die von B. Andererseits ist auch die Sequenz $A - C - B$ möglich, sollte C schneller abgeschlossen sein als B.

Die Anzahl der möglichen Sequenzen steigt im schlimmsten Fall faktoriell mit der Anzahl der parallel ausgeführten Pfade. In dem Geschäftsprozess aus Abbildung 8b werden die Pfade $B1 - B2$ sowie $C1 - C2$ parallel zueinander ausgeführt. Nachdem $B1$ vor $B2$ und $C1$ vor $C2$ ausgeführt werden muss, existieren 6 mögliche Aktivitätsreihenfolgen:

$$A - B1 - B2 - C1 - C2$$
$$A - B1 - C1 - C2 - B2$$
$$A - B1 - C1 - B2 - C2$$
$$A - C1 - C2 - B1 - B2$$
$$A - C1 - B1 - B2 - C2$$
$$A - C1 - B1 - C2 - B2$$

Dies hat zur Folge, dass die Anzahl der Sequenzen mit unterschiedlichen Aktivitätsreihenfolgen durch Nebenläufigkeit stark zunimmt, was wiederum die Bildung von Mustern erschwert.

Die Ausführungszeiten unterschiedlicher Aktivitäten können stark variieren [18]. Eine Aktivität mit langer Ausführungszeit kann einen parallelen Pfad blockieren, was die Wahrscheinlichkeit erhöht, dass bestimmte Sequenzen gebildet werden. Dies lässt sich am Beispiel aus Abbildung 8b verdeutlichen. Wird angenommen, dass die Aktivität $C1$ im Vergleich zu den anderen Aktivitäten eine deutlich längere Ausführungszeit erfordert, so blockiert diese den parallelen Pfad. Dies führt dazu, dass $C1$ und $C2$ gehäuft im hinteren Teil der Sequenzen auftreten, da der parallele Pfad mit $B1 - B2$ schneller verarbeitet wird. Die Wahrscheinlichkeit für das Auftreten einer Sequenz ist somit

abhängig von der Position von $C1$. In realen Geschäftsprozessen weisen die meisten Aktivitäten unterschiedliche Ausführungszeiten auf, daher ist zu erwarten, dass viele solcher Abhängigkeiten auftreten und Sequenzen mit bestimmten Aktivitätsreihenfolgen häufiger vorkommen. Dies begünstigt die Bildung von Mustern, insbesondere wenn eine große Anzahl an Prozessinstanzen in einem Eventlog enthalten sind. Muster, die durch die Nebenläufigkeit nur selten auftreten, werden in dieser Arbeit als Rauschen angesehen

3.4 Annahmen und Hypothesen

Das in diesem Kapitel beschriebene Konzept stützt sich auf einige Annahmen und Hypothesen, die es in dieser Arbeit zu belegen gilt:

- Die Fall- und Aktivitäts-Ids können anhand charakteristischer Merkmale der Muster von anderen Attributen im Eventlog unterschieden werden.
- Ein intelligentes System ist in der Lage diese Charakteristika für Geschäftsprozesse unterschiedlicher Größe und Komplexität zu erlernen.
- Obwohl Rauschen, Nebenläufigkeit und Zyklen die Musterbildung beeinflussen können, schränken sie die Bildung von Mustern und damit die Identifikation der Fall- und Aktivitäts-Id nur bedingt ein.

Um diese Annahmen zu belegen, wird das Konzept in Form eines Prototyps umgesetzt. Die folgenden Abschnitte beschreiben die konkreten Algorithmen und Designentscheidungen, die für dessen Implementierung getroffen werden. Die Überprüfung der Korrektheit dieser Annahmen und Hypothesen erfolgt anhand einer umfassenden Evaluation.

4 Analyse der Eventlogs

Dieses Kapitel beschreibt die konkreten Implementierungen, die zur Analyse der Eventlogs eingesetzt werden. Ziel ist die Errechnung von Kennzahlen, die als Eingangsdaten für das intelligente System verwendet werden. Der Verarbeitungsprozess eines Eventlogs kann in 3 Phasen unterteilt werden:

- **Eventlog-Konvertierung:** Dieser Schritt beinhaltet die Umwandlung der Eventlogs in ein einheitliches Datenformat.
- **Kandidatenmatrix aufstellen:** Aus allen Attributen des Eventlogs werden potenzielle Paare für Fall- und Aktivitäts-Id gebildet und als Matrix dargestellt.
- **Analyse der Kandidatenpaare:** Jedes Kandidatenpaar der Kandidatenmatrix wird auf Sequenzen und Muster untersucht. Diese werden anschließend als Basis für die Errechnung der Kennzahlen verwendet.

4.1 Konvertierung des Eventlogs

Eventlogs können aus verschiedenen Quellen stammen und in unterschiedlichen Formaten vorliegen. Das XES-Format hat sich als Standard für Eventlogs etabliert und wird von den meisten BPMS unterstützt. Allerdings kann nicht davon ausgegangen werden, dass Eventlogs immer im XES-Format vorliegen. Datenbanksysteme können Prozessdaten beispielsweise als SQL-Export bereitstellen, Softwareapplikationen halten Aktionen als Textdateien fest und Live-Dienste enthalten Schnittstellen für kontinuierliche Eventströme [16]. Damit der Prototyp auf alle Formate angewendet werden kann, müssen die Eventlogs zunächst in ein einheitliches Austauschformat konvertiert werden.

Ein Eventlog besteht aus einer Liste von Ereignissen, die wiederum aus mehreren Attributen bestehen. Jedes Attribut ist ein Schlüssel-Wert-Paar, das einen Namen und einen Wert enthält. Zusätzlich können Metainformationen im Eventlog vorhanden sein, die weitere Informationen über die Syntax, Semantik und Pragmatik der Attribute liefern [26]. Die Attribute für Fall- und Aktivitäts-Ids können beispielsweise über Metainformationen gekennzeichnet werden.

Für den Prototypen sind diese Metainformationen nicht relevant und müssen daher nicht in das Austauschformat aufgenommen werden. Dies bietet die Möglichkeit, einen Eventlog als einfache Tabelle darzustellen, wobei jede Spalte ein Ereignis und jede Zeile ein Attribut repräsentiert.

Um tabellarische zu speichern, gibt es eine Vielzahl an Formaten wie CSV, XLS (Excel), JSON oder XML. Für die Implementierung des Austauschformats wird sich aus folgenden Gründen für CSV entschieden:

- **Kompaktheit:** CSV ist ein Format mit einfacher Syntax und Struktur, das wenige strukturgebende Zeichen verwendet werden. Daher sind CSV-Dateien in der Regel kleiner im Vergleich zu anderen Formaten. Da Eventlogs oft viele Prozessinstanzen enthalten und daher große Dateien sind, trägt ein kompaktes Austauschformat dazu bei, den benötigten Ressourcenbedarf wie Arbeits- und Festplattenspeicher zu reduzieren [3].
- **Leistung:** Für große Datensätze kann CSV eine bessere Leistung im Bezug auf Lese- und Schreibvorgänge bieten als z.B. JSON oder XML. CSV-Dateien haben eine einfachere Datenstruktur und besitzen in der Regel weniger Overhead als JSON-Dateien, die verschachtelte Objekte und zusätzliche Metadaten enthalten können [3].
- **Standardbibliotheken:** CSV ist eines der Standardformate zur Speicherung tabellarischer Daten. Daher sind in jeder Programmiersprache Standardbibliotheken enthalten, die ein effizientes Lesen und Schreiben von CSV ermöglichen.

Für die Konvertierung der Eventlogs in das CSV-Format werden Python-Skripte verwendet [25]. Python ist die am weitesten verbreitete Programmiersprache für Datenanalyse und Datenmanipulation, weshalb es viele Bibliotheken gibt, die das Lesen und Schreiben verschiedener Datenformate ermöglichen [17]. Dies bietet auch die Möglichkeit, die Eventlogs vorab zu analysieren. Auf diese Weise können Informationen wie die Größe des Eventlogs oder die Anzahl der Attribute und Prozessinstanzen im Voraus erfasst werden. Im Folgenden ist ein Minimalbeispiel für die Konvertierung eines XES-Eventlogs in das CSV-Format dargestellt:

```
import pm4py
import pandas as pd

log = pm4py.read_xes("./bpi_challenge_2018.xes")
log.to_csv("./out.csv", columns=log.columns)

print("Anzahl Fall-Ids:", df['case:concept:name'].nunique())
print("Anzahl Attribute:", len(df.columns))
print("Namen der Attribute:", df.columns.tolist())
print("Anzahl der Ereignisse:", len(df))
```

Auflistung 1: Minimalskript für die Konvertierung von XES nach CSV

In der Auflistung 1 ist beispielhaft die Konvertierung eines XES-Eventlogs nach CSV zu sehen. Verwendet wird die Bibliothek *pm4py* [9], um den Eventlog einzulesen, und *pandas* [22], um die CSV-Daten zu schreiben. Darüber hinaus werden Informationen wie die Anzahl der Ereignisse, Prozessinstanzen und Attribute ausgegeben.

4.2 Aufstellen der Kandidatenmatrix

Die Analyse beginnt mit dem Einlesen eines Eventlogs, der sich in dem neutralen Austauschformat befindet. Aus den Attributen werden Paare potenzieller Fall- und Aktivitäts-Ids gebildet und anschließend zu einer Matrix geformt. Hierbei werden zwei zentrale Begriffe definiert:

Kandidatenpaare Ein *Kandidatenpaar* beschreibt eine Kombination von zwei Attributen des Eventlogs. Es kann als Tupel der Form (cId, aId) dargestellt werden, wobei cId ein Attribut ist, dass als potenzielle Fall-Id angesehen wird und aId ein Attribut ist, dass als potenzielle Aktivitäts-Id betrachtet wird. In dieser Arbeit wird zwischen zwei Arten von Kandidatenpaaren unterschieden. Das *korrekte Kandidatenpaar* ist das Kandidatenpaar, dessen Elemente der tatsächlichen Fall- und Aktivitäts-Id entsprechen und den *falschen Kandidatenpaaren*, deren Elemente von den Attributen des korrekten Kandidatenpaars abweichen.

Kandidatenmatrix Mittels Kombinatorik werden alle möglichen Kandidatenpaare aus den Attributen des Eventlogs gebildet und als Matrix dargestellt. Jedes Attribut

wird somit als potenzieller Kandidat für Fall-Id als auch Aktivitäts-Id angesehen. Innerhalb der Kandidatenmatrix befindet sich exakt ein Kandidatenpaar, das dem korrekten Kandidatenpaar entspricht. Das Problem der Identifikation der Fall- und Aktivitäts-Id in einem unlabeled Eventlog ist somit gleichbedeutend mit der Identifikation dieses Paares.

a	b	c	d
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

$$\begin{pmatrix} (a,a) & (a,b) & (a,c) & (a,d) \\ (b,a) & (b,b) & (b,c) & (b,d) \\ (c,a) & (c,b) & (c,c) & (c,d) \\ (d,a) & (d,b) & (d,c) & (d,d) \end{pmatrix}$$

Abbildung 9: Generierung der Kandidatenmatrix (rechts) aus den Attributen eines Eventlogs (links)

In der Abbildung 9 ist ein Eventlog mit den Attributen $A = \{a, b, c, d\}$ dargestellt (links). Zur Erstellung der Kandidatenmatrix wird das kartesische Produkt von $A \times A$ gebildet, dass in der rechten Abbildung zu sehen ist.

4.2.1 Kandidatenmatrix einschränken

Die Größe der Kandidatenmatrix steigt quadratisch mit der Anzahl der Attribute im Eventlog. Ein Eventlog mit vielen Attributen sorgt für eine erhebliche Steigerung der Laufzeitkomplexität, da komplexe Berechnungen für jedes Kandidatenpaar durchgeführt werden. Um die Laufzeit zu optimieren, werden Kandidatenpaare vorab ausgeschlossen, die als unwahrscheinlich oder unmöglich erachtet werden. Dies geschieht anhand von 3 Kriterien, deren Überprüfung eine geringere Laufzeitkomplexität als die späteren Berechnungen aufweisen:

Doppelte Attribute Die Fall- und Aktivitäts-Id sind immer zwei voneinander getrennte Attribute eines Eventlogs. Daher kann ein Attribut nicht gleichzeitig die Funktion der Fall-Id und Aktivitäts-Id einnehmen. Die Paare der Kandidatenmatrix, die auf der Hauptdiagonale liegen, werden aus der Matrix entfernt.

Mindestanzahl von Fall-Ids Eventlogs sind typischerweise große Dateien, die eine große Anzahl an Prozessinstanzen enthalten. Jede Prozessinstanz erzeugt eine neue,

eindeutige Fall-Id. Die Anzahl der Prozessinstanzen entspricht somit der Anzahl der eindeutigen Fall-Ids im Eventlog. Ist die ungefähre Größenordnung des Eventlogs bekannt, kann ein unterer Schwellenwert für die Anzahl der eindeutigen Fall-Ids festgelegt werden. Kandidatenpaare, deren Anzahl unter diesem Schwellenwert liegen, werden aus der Kandidatenmatrix entfernt.

Aktivitäts-Id einschränken In einem Geschäftsprozess gibt es nur eine endliche Anzahl an Aktivitäten. Ist die Größe des Prozesses abschätzbar, kann ein oberes Limit für die maximale Anzahl der Aktivitäten festgelegt werden. Alle Kandidatenpaare, die diesen Wert überschreiten, werden aus der Kandidatenmatrix entfernt.

Komplexität Die Komplexität für das Einschränken der Kandidatenmatrix ist von der Länge des Eventlogs n abhängig. Der gesamte Eventlog muss einmal durchlaufen werden, um die Anzahl der eindeutigen Werte für jedes Attribut zu bestimmen. Diese Werte können bei dem Einschränken der Kandidatenmatrix als Untergrenze verwendet werden, wenn ein Attribut als Fall-Id genutzt wird, oder als Obergrenze, wenn ein Attribut als Aktivitäts-Id verwendet wird.

Da der Eventlog nur einmal durchlaufen werden muss, um alle erforderlichen Schwellwerte zu ermitteln, beträgt die Laufzeitkomplexität $O(n)$.

4.3 Analyse der Kandidatenpaare

Die verbleibenden Kandidatenpaare der Kandidatenmatrix werden nacheinander analysiert. Für jedes Kandidatenpaar werden zunächst Sequenzen erkannt und diese anschließend zu Mustern zusammengefasst.

4.3.1 Erkennen von Sequenzen

Sequenzen werden gemäß Unterabschnitt 3.1 als Tupel der Form $(cId, [aId_1, aId_2, ..., aId_n])$ definiert. Der Pseudocode aus Algorithmus 1 zeigt einen vereinfachten Algorithmus zur Ermittlung der Sequenzen in einem Eventlog. Zunächst werden die Einträge des Eventlogs nach der potenziellen Fall-Id des Kandidatenpaars gruppiert. Hierfür durchläuft der Algorithmus den gesamten Eventlog und speichert alle Ereignisse mit derselben Fall-Id in einer Liste. Diese Listen werden nach dem Zeitstempel sortiert, um eine chronologische

Algorithmus 1 Sequenzen erkennen

```
k := Kandidatenpaar bestehend aus {cId (Fall-Id), aId (Aktivitäts-Id)}
log := Liste der Ereignisse des Eventlogs
map := Hashmap mit key=Fall-Id; value = Liste an Eventlog-Einträgen
sequences := Liste an erkannten Sequenzen
for each e ∈ log do
    if e[k.cId] ∉ map.keys then
        map.put(e[k.cId], [e])
    else
        map[e[k.cId]].append(e)
    end if
end for
for each entries ∈ map.values() do
    entries.sort()
    sequences.append({entries[0].cId, entries.map(e → e[k.aId])})
end for
```

Reihenfolge der Ereignisse herzustellen. Jeder Liste kann zu eine Sequenz konvertiert werden. Da alle Elemente einer Liste dieselbe Fall-Id besitzen, kann die Fall-Id der Sequenz aus dem ersten Element der Liste abgeleitet werden. Die Aktivitätsreihenfolge ergibt sich, indem nur die Aktivitäts-Ids aus den Ereignissen der Liste extrahiert werden.

4.3.2 Erkennen der Muster

Muster sind gemäß Unterabschnitt 3.1 Aktivitätsreihenfolgen, die gehäuft im Eventlog auftreten und als $([a_1, a_2, \ldots, a_m], [c_1, c_2, \ldots, c_n])$ definiert sind. Eine vereinfachte Implementierung der Mustererkennung ist in Algorithmus 2 dargestellt. Zunächst wird eine leere Liste aller erkannten Muster initialisiert. Für jede Sequenz wird überprüft, ob bereits ein Muster mit derselben Aktivitätsreihenfolge existiert. Ist dies nicht der Fall, wird mittels der Fall-Id und Aktivitätsreihenfolge der Sequenz ein neues Muster erzeugt und in die Liste aller Muster aufgenommen. Existiert bereits ein Muster mit derselben Aktivitätsreihenfolge, so wird lediglich die Fall-Id zu diesem Muster hinzugefügt. Dieser Prozess wird fortgesetzt, bis alle Sequenzen des Kandidatenpaars durchlaufen sind. Die Mustererkennung für das Kandidatenpaar ist somit abgeschlossen, und die Verarbeitung des nächsten Kandidatenpaars kann beginnen.

Algorithmus 2 Muster erkennen

```
sequences := Liste an erkannten Sequenzen
patterns := Liste an erkannten Muster
exists := Ob bereits ein Muster zu der aktuellen Sequenz existiert
for each sequence ∈ sequences do
    exists = false
    for each pattern ∈ patterns do
        if sequence == pattern.sequence then
            exists = true
            pattern.addCId(sequence.cId)
        end if
    end for
    if exists == false then
        patterns.append({sequence : sequence, cIds : [sequenze.cId]})
    end if
end for
```

4.3.3 Komplexität der Sequenz und Musteranalyse

Die Kandidatenmatrix besteht aus $|A|^2$ Zellen, wobei $|A|$ die Anzahl der Attribute im Eventlog ist. Jedes Kandidatenpaar wird auf Sequenzen und Muster untersucht. Dieser Prozess lässt sich stark parallelisieren, da die Analyse eines Kandidatenpaars keine Seiteneffekte auf andere Kandidatenpaare hat und somit nebenläufig sein kann. Die Suche nach Sequenzen erfordert das einmalige Durchlaufen des gesamten Eventlogs, dessen Länge mit n definiert ist. Die Sequenzanalyse aller Kandidatenpaare beträgt somit $O(|A|^2 \cdot n)$ bzw. $O(n)$, wenn alle Kandidatenpaare nebenläufig verarbeitet werden.

Für die Identifikation der Muster müssen die Sequenzen untereinander verglichen werden. Wenn s die Anzahl der erkannten Muster ist, ergibt sich eine Komplexität von s^2 für den Vergleich der Muster untereinander. Im günstigsten Fall hinischtlich der Komplexität besteht ein Eventlog nur aus einer einzelnen langen Sequenz, wodurch nur ein Muster ($s = 1$) erkannt wird. In diesem Fall steigt die Komplexität für Sequenz- und Mustererkennung linear mit der Größe des Eventlogs, sofern eine ausreichende Parallelisierung vorhanden ist. Im Worst-Case-Szenario bilden sich nur Sequenzen der Länge 1. Dies tritt auf, wenn beide Attribute eines Kandidatenpaars Id-Felder sind und nur einzigartige Werte enthalten. In diesem Fall entspricht die Anzahl der Muster s der Länge des Eventlogs n. Dies führt zu einer quadratischen Komplexität

In Tabelle 3 sind die Gesamtkomplexitäten der Kandidatenpaaranalyse unter Berück-

sichtigung des Worst- und Best-Case-Szenarios sowie des Einflusses der Nebenläufigkeit (*Parallel*) dargestellt. Ausschlaggebend für die reale Laufzeitkomplexität ist die Größe des Eventlogs n, da diese um ein Vielfaches größer ist als die Anzahl der Attribute im Eventlog. Das Worst-Case-Szenario tritt auf, wenn in einem Eventlog 2 Attribute existieren, die nur einzigartige Werte enthalten. Um die Bildung dieser Kandidatenpaare zu vermeiden, können diese mittels der in Unterunterabschnitt 4.2.1 vorgestellten Einschränkung vorab ausgeschlossen werden.

	Best-Case	Worst-Case
Normal	$O(\|A\|^2 \cdot n)$	$O(\|A\|^2 \cdot n^2)$
Parallel	$O(n)$	$O(n^2)$

Tabelle 3: Zusammengefasste Komplexität der Sequenz und Mustererkennung im Worst-/Best-Case Szenario

4.3.4 Berechnen der Kennzahlen

Mittels der gefundenen Muster lassen sich Kennzahlen für jedes Kandidatenpaar errechnen. Diese spiegeln die Charakteristiken der erkannten Muster in gebündelter Form wider und ermöglichen somit eine Bewertung der Plausibilität der einzelnen Kandidatenpaare. Im Prototyp werden 6 unterschiedliche Kennzahlen verwendet:

Anzahl Fall-Ids Es wird bestimmt, wie viele einzigartige Fall-Ids für ein Kandidatenpaar existieren. Dieser Wert beschreibt, wie viele Prozessinstanzen sich in dem Eventlog befinden. Die meisten Prozess-Mining-Verfahren benötigen eine ausreichend große Datengrundlage, um effektiv verwendet werden zu können. Daher wird erwartet, dass die Anzahl der einzigartigen Fall-Ids hoch ist. Zur Ermittlung dieser Kennzahl wird der Eventlog einmal durchlaufen und gezählt, wie viele unterschiedliche Fall-Ids existieren.

Anzahl Aktivitäts-Ids Es wird bestimmt, wie viele einzigartige Aktivitäts-Ids für das Kandidatenpaar existieren. Dieser Wert gibt an, wie viele unterschiedliche Aktivitäten im Prozess enthalten sind und ist damit ausschlaggebend für dessen Größe. Zur Ermittlung dieser Kennzahl wird der Eventlog einmal durchlaufen und gezählt, wie viele unterschiedliche Aktivitäts-Ids existieren.

Anzahl Muster Es wird die Anzahl der Muster für ein Kandidatenpaar bestimmt. Dieser Wert entspricht der Anzahl eindeutiger Pfade die in dem Geschäftsprozess vom Start- zum Endknoten führen.

Musterhäufigkeit Diese gibt an, wie häufig die erkannten Muster im Durchschnitt vorkommen. Jeder Pfad durch den Geschäftsprozess entspricht einem erkannten Muster. Wird der Prozess mehrfach entlang desselben Pfades durchgeführt, verursacht dies eine Wiederholung des zugehörigen Musters. In einem Eventlog, welcher viele Prozessinstanzen enthält, ist daher eine hohe Musterhäufigkeit zu erwarten.

Errechnet wird die Musterhäufigkeit über die Formel:

$$\frac{\sum_{i=1}^{|m|} |m_i.cIds|}{|m|}$$

Jedes Muster m_i enthält eine Liste an Prozessinstanzen $m_i.cIds$, welche das Muster ausgelöst haben. Es werden die Längen dieser Listen für alle Muster eines Kandidatenpaars bestimmt und aufsummiert. Um die durchschnittliche Musterhäufigkeit zu erhalten, wird diese durch die Anzahl der erkannten Muster $|m|$ geteilt.

Musterlänge Diese gibt die durchschnittliche Anzahl der Aktivitäten an, die eine Prozessinstanz bei der Durchführung des Geschäftsprozesses durchläuft.

$$\frac{\sum_{i=1}^{|m|} |m_i.aIds| \cdot |m_i.cIds|}{\sum_{j=1}^{|m|} |m_j.cIds|}$$

Jedes Muster m_i besitzt eine Aktivitätsreihenfolge $m_i.aIds$ und eine Liste an Fall-Ids $m_i.cIds$. Damit die Länge der Muster, die häufiger im Eventlog auftreten, eine höhere Gewichtung erhalten, wird die Musterlänge mit der Anzahl der Vorkommnisse des Musters $|m_i.aIds|$ multipliziert. Dieser Wert wird für alle Muster eines Kandidatenpaars gebildet und aufsummiert. Die durchschnittliche Musterlänge errechnet sich aus dem Verhältnis dieser Summe zu der absoluten Häufigkeit aller Muster.

Musterähnlichkeit Die Musterähnlichkeit gibt an, wie sehr sich die Muster eines Kandidatenpaars untereinander gleichen. Ein Geschäftsprozess besitzt immer einen Start- und Endknoten. Alle Pfade durch den Prozess beginnen bzw. enden an diesen Knoten. Bedingungsknoten spalten einen Pfad in mehrere Subpfade auf. Da alle Pfade am Startknoten beginnen, weisen alle Subpfade die gleiche Aktivitätsreihenfolge vom Startknoten bis zum ersten Bedingungsknoten auf. Analog dazu gilt dasselbe Prinzip, wenn die Subpfade wieder vereinigt werden. Von der Vereinigung bis zum nächsten Bedingungs- oder Endknoten besitzen diese ebenfalls dieselbe Aktivitätsreihenfolge. Daher ist zu erwarten, dass die Aktivitätsreihenfolgen der Pfade in einem Geschäftsprozess eine hohe Ähnlichkeit zueinander aufweisen.

Der Prototyp enthält einen optimierten Algorithmus, der nur die Musterähnlichkeit am Anfang und Ende der Aktivitätsreihenfolgen der Muster bestimmt. Dies sorgt für eine Reduktion der Laufzeitkomplexität, da bei der ersten Abweichung der Aktivitätsreihenfolge abgebrochen wird. Allerdings reduziert dies den Informationsgehalt, da die Ähnlichkeit der Aktivitätsreihenfolgen in der Mitte der Muster ignoriert wird.

Algorithmus 3 Musterähnlichkeit

```
m := Liste an Mustern eines Kandidatenpaars
patternSimilarity := Kumulierte Summe der Musterähnlichkeit
exists := Ob bereits ein Muster zu der aktuellen Sequenz existiert
for each m_i ∈ m do
    for each m_j ∈ m do
        similarity = 0
        length = min(|m_i.aIds|, |m_j.aIds)
        for a = 0 to length do
            if m_i.aIds[a] == m_j.aIds[a] then
                similarity += 1
            else
                break
            end if
        end for
        similaritySum+ = similarity
    end for
end for
patternSimilarity = similaritySum/(|m|^2)
```

In Algorithmus 3 ist beispielhaft dargestellt, wie die Musterähnlichkeit am Anfang der Muster errechnet wird. Es werden nacheinander die erkannten Muster m eines Kandidatenpaars miteinander verglichen. Für jede Musterkombination (m_i, m_j) werden die

Aktivitätsreihenfolgen $m_i.aIds$ und $m_j.aIds$ der Muster untersucht. Es wird ab dem ersten Eintrag der Listen bestimmt, wie viele Einträge von $m_i.aIds$ identisch mit $m_j.aIds$ sind. Bei der ersten abweichenden Aktivität wird der Vergleich abgebrochen. Die Anzahl der identischen Einträge wird als Ähnlichkeit (*similarity*) der beiden Muster m_i und m_j definiert. Die allgemeine Musterähnlichkeit eines Kandidatenpaars (*patternSimilarity*) ergibt sich aus der durchschnittlichen Ähnlichkeiten aller miteinander verglichenen Muster.

Ebenfalls im Protoyp implementiert ist ein Algorithmus zur Bestimmung der Musterähnlichkeit am Ende der Muster. Dieser ist nahezu identisch zu Algorithmus 3, allerdings wird die Aktivitätsreihenfolge von $m_i.aIds$ und $m_j.aIds$ vor dem Vergleich umgekehrt. Somit werden die Aktivitäten ausgehend vom Ende des Prozesses miteinander verglichen. Die allgemeine Musterähnlichkeit eines Kandidatenpaars ergibt sich aus dem Durchschnitt der Musterähnlichkeit am Start und am Ende der Muster.

4.4 Rauschunterdrückung

Reale Eventlogs aus Produktivsystemen enthalten Rauschen [13]. Rauschen sorgt gemäß Unterabschnitt 3.3 zur Bildung einer großen Anzahl von Mustern, die alle eine geringe Häufigkeit aufweisen. Dies hat Einfluss auf die errechneten Kennzahlen, da diese oft auf der Musteranzahl und Musterhäufigkeit basieren und somit verfälscht werden.

Um dem entgegenzuwirken, enthält der Prototyp einen Mechanismus zur Identifikation und Beseitigung des Rauschens. Dieser besteht darin, dass Muster, die eine geringe Häufigkeit aufweisen, für die Berechnung der Kennzahlen ausgeschlossen werden. Für alle Kennzahlen existiert neben der normalen eine bereinigte Variante, die nur auf der Teilmenge der am häufigsten vorkommenden Muster basiert. Somit wird die Anzahl der errechneten Kennzahlen auf 12 verdoppelt.

Um diese Teilmenge zu bestimmen, werden zunächst alle Muster m nach ihrer Häufigkeit sortiert. Der Parameter *noisePercentage* ist im Prototypen definiert und gibt an, wie viel Prozent der Muster als Rauschen angesehen werden. Aus den sortierten Mustern m wird das Muster an der Stelle $noisePercentage \cdot |m|$ ausgewählt. Alle Muster, welche eine größere Häufigkeit als dieses aufweisen, werden in die Teilmenge mit aufgenommen. Wenn beispielsweise $noisePercentage = 0.5$ ist, werden mindestens die Hälfte der erkannten Muster als Rauschen erachtet und ignoriert.

5 Trainingsbestand generieren

Das in Abschnitt 3 vorgestellte Konzept verwendet ein intelligentes System für die Auswertung der in Unterunterabschnitt 4.3.4 errechneten Kennzahlen. Eine der großen Herausforderungen in Bezug auf intelligente Systeme ist der große Datenbedarf, der für das Anlernen und die Evaluation der Modelle erforderlich ist [5]. Insbesondere im Bereich des Process-Minings ist die Menge der frei zugänglichen Datensätzen gering, da Geschäftsprozesse von Unternehmen in der Regel als kritische Infrastruktur und Firmeneigentum angesehen werden [11]. Daher sind Unternehmen selten bereit, diese Informationen preiszugeben [11]. Die öffentlich zugänglichen Datensätze sind meist unvollständig oder bilden nur einen kleinen Auszug aus den Systemen ab [11].

Darüber hinaus beschäftigen sich Prozess-Mining-Verfahren primär mit der Gewinnung von strukturellen Prozessinformationen aus Eventlogs, wie beispielsweise Process Discovery, Conformance-Checking oder Process-Enhancement [1, 16]. Viele der frei zugänglichen Datensätze sind auf diesen Anwendungsfall ausgelegt und enthalten oftmals nur die Attribute für Fall-Id, Aktivitäts-Id und einen Erstellungszeitstempel [8, 16]. Diese Felder sind ausreichend, um die Struktur eines Prozesses abzubilden (siehe Unterabschnitt 2.2). Das in dieser Arbeit beschriebene Konzept hebt sich von den oben genannten Verfahren ab, da neben den strukturgebenden Feldern auch weitere Attribute des Eventlogs benötigt werden. Dies schränkt die Anzahl der geeigneten Datensätze stark ein. Eventlogs, die für diese Arbeit als geeignet angesehen werden, müssen folgende Eigenschaften aufweisen:

- Viele Prozessinstanzen enthalten, da erst dies zur Bildung von Mustern führt.
- Viele Attribute enthalten, da diese die Größe der Kandidatenmatrix definieren. Dies sorgt für die Bildung einer Vielzahl falscher Kandidatenpaare. Somit wird die Wahrscheinlichkeit verringert, dass das intelligente System per Zufall ein richtiges Ergebnis liefert.
- Verschiedene Arten von Attributen aufweisen, die sich in ihrem Verhalten unterscheiden, wie bspw. boolesche Flags, Id-/Status-Felder oder Zeitstempel. Dies führt zur Bildung von Kandidatenpaaren mit stärker abweichenden Charakteristiken. Somit entsteht ein umfangreicherer Datenbestand, der es dem intelligenten System ermöglicht, komplexere Szenarien zu verstehen.

Eine alternative zu freien Datensätzen, ist die Verwendung von synthetischen Eventlogs. Synthetische Eventlogs stammen nicht aus realen Prozessen, sondern werden über Generatoren künstlich erzeugt. Dies bietet den Vorteil, dass der Umfang und die Größe des Datenbestands beliebig angepasst und erweitert werden kann [12]. Zeigt das intelligente System beispielsweise ein Defizit bei einer bestimmten Art von Geschäftsprozessen, so können diese zusätzlich generiert und in den Trainingsdatensatz mit aufgenommen werden.

Aus diesen Gründen, wird in dieser Arbeit ein Prozessgenerator entwickelt, der für die Bereitstellung der Trainingsdaten verwendet wird. In diesem Kapitel wird dessen Aufbau und Funktionsweise vorgestellt.

5.1 Anforderungen an den Prozessgenerator

Die Güte eines intelligenten Systems hängt stark von der Qualität der Trainingsdaten ab [5]. Daher ist es von großer Bedeutung, dass die Geschäftsprozesse des Prozessgenerators möglichst praxisnah und unvoreingenommen sind. Ist dies nicht der Fall, so könnte das intelligente System anstatt der Identifikation der Fall- und Aktivitäts-Id charakteristische Merkmale und Tendenzen des Generators erlernen. Diese Problemstellung wird gemäß Absatz 2.3.1 als Voreingenommenheit der Stichproben bezeichnet. Um dies zu verhindern, muss der Prozessgenerator bestimmte Anforderungen erfüllen:

- **Unvoreingenommen:** Die Ausgabe des Generators sollte möglichst unvoreingenommen (unbiased) gegenüber einer bestimmten Art von Geschäftsprozessen sein. Wenn alle möglichen Geschäftsprozesse unter einer gegebenen Konfiguration als Datenpunkte in einem mehrdimensionalen Raum dargestellt werden, soll der Generator in der Lage sein, diesen Raum sowohl vollständig abzubilden, ohne dass Hotspots entstehen. Anders ausgedrückt, die Generierung der Prozesse sollte so gleichverteilt wie möglich erfolgen, um keine Präferenzen gegenüber einer bestimmten Art von Prozessen zu entwickeln. Dies soll verhindern, dass das intelligente System anstelle der Interpretation der Kennzahlen diese Abhängigkeiten erlernt.
- **Konfigurierbarkeit:** Der Generierungsprozess muss hochgradig konfigurierbar sein. Dies ermöglicht die gezielte Erzeugung verschiedener Szenarien, wie beispielsweise kleine/große oder einfache/komplexe Geschäftsprozesse. Die Konfigurierbarkeit ist

entscheidend, um eine breit gefächerte Datengrundlage für die Validierung zu schaffen. Dies soll dazu führen, dass das intelligente System auch auf realen Prozessen beliebiger Größe und Komplexität anwendbar ist.

- **Konsistenz:** Die generierten Prozesse müssen in ihrer Struktur konsistent und korrekt sein. Alle Einschränkungen und Regeln, denen reale Geschäftsprozesse unterliegen, müssen eingehalten werden.
- **Deterministisch:** Die Generierung der Prozesse muss deterministisch sein. Einerseits erleichtert dies die Entwicklung des Prototyps, da auftretende Fehler reproduzierbar sind. Andererseits ermöglicht dies die Replikation der Ergebnisse, was für das wissenschaftliche Arbeiten von großer Bedeutung ist.

5.2 Abgrenzung zu PLG

In [12] wird bereits auf die unzureichende Anzahl frei zugänglicher Testdaten im Prozess-Mining-Bereich eingegangen. Die Forschenden entwickeln die Java-Anwendung PLG (*Process and Log Generator*), die in der Lage ist, zufällige Geschäftsprozesse zu generieren und zu simulieren. In der weiterführenden Arbeit [11] wird die Anwendung um die Anwendungsfälle von Concept-Drift, Streaming-Events und Rauschen erweitert. PLG soll Forschenden ermöglichen, ihre Prozess-Mining-Algorithmen und -Konzepte umfangreich zu validieren, auch wenn keine ausreichende Datengrundlage existiert.

PLG ist auf die klassischen Anwendungsfälle wie Process Discovery oder Conformance Checking ausgelegt. Die Identifikation von Fall- und Aktivitäts-Ids, die in dieser Arbeit behandelt wird, unterscheidet sich von diesen Standardfällen. Die Struktur der Prozesse spielt nur eine sekundäre Rolle. Wichtiger sind die Attribute, insbesondere deren Wertverlauf und Wertwiederholungen. Die Möglichkeiten von PLG, das Verhalten von Attributen zu beeinflussen, sind begrenzt, da diese für klassische Prozess-Mining-Verfahren keine große Bedeutung haben [12, 11].

Zudem existiert bislang keine Literatur, die die Unvoreingenommenheit der von PLG erzeugten Geschäftsprozesse prüft. Es ist nicht auszuschließen, dass die generierten Prozesse bestimmte Sequenzen oder Muster bevorzugen [12, 11].

Zuletzt bietet PLG nur eingeschränkte Möglichkeiten, gezielt die Größe und Komplexität der generierten Prozesse zu definieren. Diese werden primär über die Tiefe des Prozesses und die Wahrscheinlichkeit des Auftretens verschiedener Knotentypen gesteuert. Soll

beispielsweise ein kleiner Prozess generiert werden, so wird die Wahrscheinlichkeit, mit der der nächste Knoten dem Endknoten entspricht, auf einen hohen Wert gesetzt. Dies verhindert jedoch nicht, dass durch diese Konfiguration auch lange Prozesse zufällig erzeugt werden. Ein gezielter Aufbau eines Datenbestands, der nur spezifische Szenarien enthält, wird somit erschwert. Auch kann dies zu Ungenauigkeiten bei der Validierung führen, wenn ein Szenario isoliert betrachtet wird, da nicht sichergestellt werden kann, dass alle Prozesse diesem Szenario entsprechen [12, 11].

In dieser Arbeit wird sich daher gegen den Einsatz von PLG als Prozessgenerator entschieden. Stattdessen wird eine eigene Prozess-Engine und ein Generator für zufällige Geschäftsprozesse entwickelt, die beide auf den Anwendungsfall dieser Arbeit spezialisiert sind. Konzepte, wie der Aufbau der Geschäftsprozesse und das Erzeugen von Rauschen, sind an die Implementierung von PLG angelehnt [12, 11].

PLG wird dennoch als zweite Instanz zur Validierung des Prototyps verwendet. Die Prozesse der entwickelten Prozess-Engine sind unzureichend, um die Genauigkeit des intelligenten Systems zu bestimmen, da diese auch für den Trainingsprozess verwendet werden. Daher werden Prozesse aus einem zweiten, unabhängigen Generator wie PLG für die Validierung hinzugezogen, um die Robustheit der Ergebnisse zu erhöhen [12, 11].

5.3 Die Prozess-Engine

Die Prozess-Engine ermöglicht es, Geschäftsprozesse zu definieren und zu simulieren. Das Ergebnis dieser Simulation sind Eventlogs, die als Trainingsbestand für das intelligente System verwendet werden. Die Prozess-Engine wird von dem Prozessgenerator als Basis verwendet, um zufällige prozedurale Geschäftsprozesse zu erzeugen.

5.3.1 Aufbau

Ein Geschäftsprozess kann als Abhängigkeitsgraph dargestellt werden, der aus Knoten und Kanten besteht [11]. Elemente eines Geschäftsprozesses, wie Aktivitäten, Entscheidungen oder Subprozesse, können als Knoten in diesem Graphen angesehen werden. Dabei unterscheiden sich die Knoten in ihrem Verhalten bei der Ausführung. Aktivitäten erzeugen beispielsweise Einträge im Eventlog, während Bedingungen aus einer Liste an Knoten den nächsten Nachfolger auswählen.

Die Kanten symbolisieren die Verbindungen zwischen diesen Knoten. Diese Verbindungen werden implementiert, indem jeder Knoten seinen direkten Nachfolger kennt. Dies ermöglicht ein einfaches Einfügen und Löschen von Knoten, da nur die Referenzen auf die Nachfolgerknoten an den entsprechenden Stellen angepasst werden müssen.

Die Simulation eines Prozesses entspricht dem vollständigen Durchlaufen des Abhängigkeitsgraphen. Bei der Simulation wird der erste Knoten des Graphen ausgeführt. Iterativ werden daraufhin alle Nachfolgeknoten ermittelt und ausgeführt. Dieser Vorgang wird fortgesetzt, bis kein weiterer Nachfolgeknoten mehr vorhanden oder der Endknoten erreicht ist.

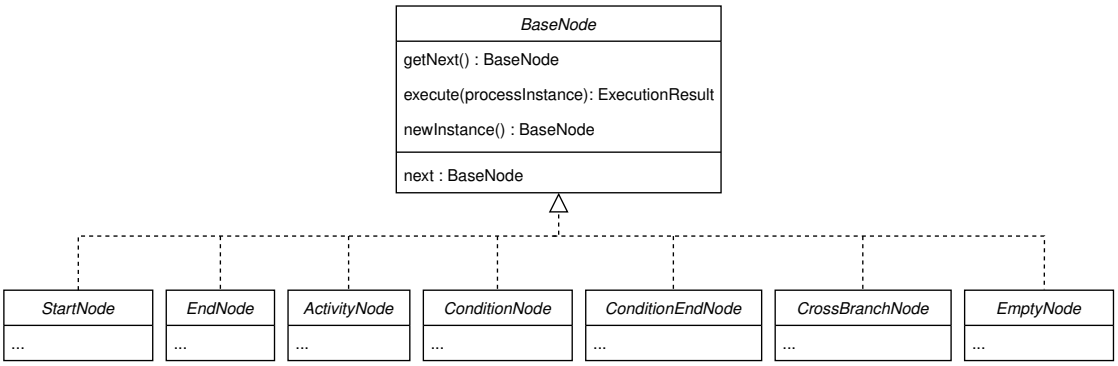


Abbildung 10: Klassendiagramm der implementierten Prozessknoten.

In dem vereinfachten Klassendiagramm aus Abbildung 10 wird die grundlegende Implementierung eines Knotens dargestellt. Jeder Knoten basiert auf einem Grundelement, dem Basis-Knoten (*BaseNode*). Dieser enthält eine Referenz auf den nächsten Knoten (*getNext*) und definiert die Methoden *execute* und *newInstance*, die von allen abgeleiteten Knoten implementiert werden. Die *execute*-Methode wird während der Ausführung des Knotens aufgerufen und bestimmt dessen Verhalten. Beispielsweise sorgt der Endknoten bei der Ausführung für die Beendigung des Prozesses oder eine Aktivität erzeugt einen Eintrag im Eventlog. Die Methode *newInstance* ist für das Erstellen einer Kopie des Knotens zuständig. Alle Zustände, die der Knoten enthält, werden in der Kopie auf ihre Standardwerte zurückgesetzt. Dies erlaubt es, eine Kopie des gesamten Prozesses zu erstellen, um beispielsweise eine neue Prozessinstanz zu simulieren.

5.3.2 Knotentypen

In Abbildung 10 sind alle Implementierten Knotentypen dargestellt. Im Folgenden wird deren Funktion beschrieben:

Start-/Endknoten (*StartNode/EndNode*) Diese Knoten kennzeichnen den Beginn bzw. das Ende des Prozesses. Ein Prozess muss immer mit dem Startknoten beginnen und beim Endknoten enden. Diese Einschränkung ermöglicht es der Prozess-Engine, ungültige Prozessdefinitionen zu identifizieren, wie beispielsweise Pfade, die ins Leere laufen.

Aktivitätsknoten (*ActivityNode*) Ein Aktivitätsknoten repräsentiert eine Aktivität innerhalb des Prozesses. Es ist möglich, Attribute an einen Aktivitätsknoten anzuhängen. Bei der Definition des Knotens werden der Name der Aktivität, eine durchschnittliche Ausführungsdauer und die Standardabweichung von dieser angegeben. Bei der Ausführung erstellt der Aktivitätsknoten einen Eintrag im Eventlog. Informationen wie die Fall-Id, der Name der Aktivität (Aktivitäts-Id), die errechnete Ausführungsdauer und die Werte der verknüpften Attribute werden dabei im Eventlog festgehalten.

Bedingungsknoten (*ConditionNode*) Dieser Knoten steht für eine Bedingung, die einen Pfad im Prozess in mehrere Subpfade aufteilt. Ein Bedingungsknoten enthält intern eine Liste potenzieller Folgeknoten, die mit einer Auftrittswahrscheinlichkeit verknüpft sind. Bei der Ausführung eines Bedingungsknotens wird zufällig gemäß dieser Wahrscheinlichkeit ein Knoten dieser Liste ausgewählt und als Folgeknoten verwendet. Zu einer Bedingung im Prozess gehört immer ein Bedingungsendknoten (*ConditionEndNode*). Alle Pfade einer Bedingung müssen an diesem wieder vereint werden.

Sprungknoten (*CrossBranchNode*) Ein spezieller Knoten, der das Ende eines Sprungpfades markiert. Dieser verweist auf einen beliebigen Knoten im Prozess, der auch vor bzw. nach dem Sprungknoten oder auf einem parallelen Subpfad liegen kann. Auf Sprungpfade wird gesondert in Unterunterabschnitt 5.3.3 eingegangen.

Platzhalterknoten (*EmptyNode*) Ein Knoten im Prozess, der keine Logik enthält. Dieser dient zur Fehlersuche oder als Platzhalter für Knoten, die noch nicht erstellt sind. Soll ein Sprungknoten beispielsweise auf einen Knoten verweisen, der noch nicht im Prozess enthalten ist, zeigt dieser bis zu dessen Erstellung auf einen Platzhalterknoten.

5.3.3 Sprungpfade

Die Implementierung der Bedingungen unterliegen einer grundlegenden Einschränkung. Eine *ConditionNode* teilt einen Eingangspfad in mehrere Subpfade auf. Diese Subpfade müssen zu einem späteren Zeitpunkt im Prozess an einem *ConditionEndNode* wieder vereinigt werden. Dies führt dazu, dass nur lineare Prozessflüsse abgebildet werden können. In einem linearen Prozess existiert nur eine endliche Anzahl an Pfaden, die vom Start- zum Endknoten führen. Jede ausgeführte Aktivität trägt zum Fortschritt des Prozesses bei.

Allerdings ermöglicht diese Designentscheidung das Berechnen komplexer Statistiken, die die Form der Prozesse widerspiegeln. Beispielsweise kann an jeder Stelle des Prozesses die Prozesstiefe bestimmt werden oder die Länge der Subpfade einer Bedingung gemessen werden.

Um die Prozesstiefe für einen Knoten zu bestimmen, werden alle *ConditionNodes* gezählt, die vor diesem Knoten liegen. Eine *ConditionNode* erzeugt einen Subpfad und erhöht damit die Prozesstiefe. Durch die *ConditionEndNodes* ist bekannt, wann ein Subpfad endet. Daher kann von der Anzahl der *ConditionNodes* die Anzahl der *ConditionEndNodes*, die sich vor einem Knoten befinden, abgezogen werden, um die Prozesstiefe zu ermitteln.

Ebenso kann die Länge der Subpfade bestimmt werden. Diese ergibt sich aus der Anzahl der Knoten, die zwischen einem *ConditionNode* und dem dazugehörigen *ConditionEndNode* liegen.

Bedingungen in realen Geschäftsprozessen folgen allerdings nicht der Einschränkung, dass alle Pfade einer Bedingung wieder vereinigt werden müssen. Ein Beispiel hierfür sind Fehlerroutinen. Tritt ein Fehler auf, wird eine definierte Abfolge an Aktivitäten ausgelöst, die diesen Fehler behandeln. Es können mehrere Bedingungen im Prozess existieren, die zu unterschiedlichen Zeitpunkten diesen Fehler prüfen und alle auf dieselbe Aktivitätsreihenfolge verweisen. Ein weiteres Beispiel sind Zyklen und Revisionen. Nach Durchführung einer Reihe von Aktivitäten wird geprüft, ob das Ergebnis valide ist. Bei

einem negativen Resultat werden die Aktivitäten erneut ausgeführt. Solche Szenarien, wie das Verzweigen eines Pfades auf einen anderen Prozesspfad im ersten Beispiel oder das Entstehen von Zyklen durch Verweis auf bereits durchlaufene Knoten im zweiten Beispiel, lassen sich nicht mit den Bedingungen der entwickelten Prozess-Engine abbilden. Um diese Verhaltensweisen darstellen zu können, wird das Konzept der *Sprungpfade* entwickelt.

Sprungpfade repräsentieren in der Prozess-Engine spezielle Pfade einer Bedingung, die nicht wieder mit den anderen Pfaden derselben Bedingung zusammengeführt werden. Jeder Sprungpfad endet stets mit einem *CrossBranchNode*. Dieser Knoten verweist auf eine beliebige Stelle im Geschäftsprozess, die auch vor oder nach der Bedingung oder auf einem parallelen Subpfad liegen kann. Der sonst lineare Prozessfluss wird dadurch unterbrochen. In einer Bedingung mit n Pfaden können maximal $n-1$ Sprungpfade existieren. Diese Einschränkung gewährleistet, dass der Endknoten immer erreichbar ist, da mindestens ein Pfad eine durchgängige Verbindung vom Start zum Endknoten enthält.

Sprungknoten bieten zudem 2 weiteren Vorteil. Sie erlauben eine Unterscheidung zwischen regulären Pfaden und Sprungpfaden. Bei der Erstellung der zufälligen Geschäftsprozesse ermöglicht diese eine gezielte Konfiguration der „Linearität“. Es kann beispielsweise angegeben werden, dass es sich bei 20 Prozent aller Pfade einer Bedingung um Sprungpfade handeln soll. Zusätzlich vereinfacht dies die Identifikation von endlosen Zyklen, da diese nur an Sprungpfaden entstehen können.

5.3.4 Die Prozesssimulation

Mit der entwickelten Prozess-Engine ist es möglich, Prozessinstanzen aus einem Prozess zu erstellen und diese anschließend zu simulieren.

In Algorithmus 4 wird das grundlegende Vorgehen der Simulation einer Prozessinstanz dargestellt. Bei der Initialisierung der Simulation wird eine Konfiguration ($config$) angegeben, welche die Anzahl der Prozessinstanzen, den prozentualen Anteil von Rauschen und einen Startzeit-Offset ($startOffset$) enthält. Jede Prozessinstanz besitzt eine eigene Modellzeit ($simulationTime$) in Form einer fortlaufenden Nummer. Bei der Ausführung eines Knotens wird diese Modellzeit um dessen Ausführungszeit inkrementiert. Der Startzeit-Offset ermöglicht es, dass Prozessinstanzen mit einem zufälligen Abstand zueinander starten. Initial wird die Startzeit der Prozessinstanz zufällig zwischen 0 und diesem

Algorithmus 4 Simulation einer Prozessinstanz

```
config := Konfiguration der Simulation
piData := Datenobjekt der Prozessinstanz
bpmn := Der Geschäftsprozess

bpmnCopy = bpmn.newInstance()
piData.currentNode = bpmnCopy.firstNode
piData.simulationTime = random(0, config.startOffset)
while piData.currentNode do
    currentNode = piData.currentNode
    currentNode.execute(piData)
    applyNoise(piData, config)
    piData.currentNode = currentNode.nextNode
end while
```

Offset gesetzt. Jede Prozessinstanz besitzt darüber hinaus ein Datenobjekt (*piData*), welches Informationen wie beispielsweise die Einträge des Eventlogs, den aktuellen Knoten, die Modellzeit der Instanz und Statistiken über die getätigten Aktionen enthält.

Bei der Simulation einer Prozessinstanz wird zunächst eine unabhängige Kopie (*bpmnCopy*) des Prozesses erstellt. In *piData* wird als aktiver Knoten (*currentNode*) der erste Knoten des kopierten Prozesses gesetzt. Der aktive Knoten wird ausgeführt und erhält dabei Zugriff auf das Datenobjekt der Prozessinstanz. Dies erlaubt es dem Knoten, beispielsweise Eventlog-Einträge zu erzeugen, die Modellzeit zu modifizieren oder Statistiken zu beeinflussen.

Nach der Ausführung des Knotens wird der Generator für Rauschen aufgerufen. Dieser fügt zufällig, gemäß den Wahrscheinlichkeiten aus *config*, Rauschen hinzu. Im folgenden Unterunterabschnitt 5.3.5 werden die Arten von Rauschen, die der Generator erzeugen kann, beschrieben. Der Folgeknoten des aktiven Knotens wird daraufhin als neuer aktiver Knoten gesetzt und ausgeführt. Dies wird wiederholt, bis der letzte Knoten des Prozesses durchlaufen ist und daher kein Folgeknoten mehr existiert. Die Simulation der Prozessinstanz ist damit abgeschlossen.

Da die Prozessinstanzen alle über eine eigene Kopie des Prozesses sowie ein eigenes Datenobjekt verfügen, lässt sich die Simulation parallelisieren. Abhängig von der Anzahl der Prozessinstanzen kann dies die Laufzeit der Simulation stark verkürzen. Nach Beendigung aller Instanzen werden die Objekte der Prozessinstanzen gesammelt und zu übergeordneten Simulationsobjekten zusammengefasst. Der vollständige Eventlog ergibt

sich beispielsweise durch das Zusammenfügen aller Eventlog-Einträge der Instanzen. Für die Statistiken werden Durchschnitts-, Minimal- und Maximalwerte über alle Instanzen gebildet.

5.3.5 Der Rausch-Generator

In Unterabschnitt 3.3 wird bereits auf Rauschen in Eventlogs eingegangen. Damit die Eventlogs der simulierten Prozesse möglichst realistisch sind, enthält die Prozess-Engine einen Generator für Rauschen. Dieser erlaubt es, verschiedene Arten von Rauschen zu erzeugen. Gemäß [11] kann Rauschen in Eventlogs auf drei Ebenen auftreten:

- **Ebene der Prozessinstanz:** Der Ablauf und die Aktivitätsreihenfolge der Ereignisse sind fehlerhaft. Beispielsweise werden Aktivitäten doppelt erfasst oder es fehlt der Beginn oder das Ende des Prozesses.
- **Ebene eines Ereignisses:** Einzelne Ereignisse sind falsch in der Prozessinstanz eingeordnet. Ein Ereignis enthält beispielsweise eine falsche Aktivitäts-Id oder der Zeitpunkt des Ereignisses ist falsch bestimmt worden, was die Reihenfolge der Ereignisse beeinflussen kann.
- **Datenebene:** Die nicht strukturgebenden Felder eines Ereignisses sind fehlerhaft. Die Attribute enthalten beispielsweise invalide oder inkonsistente Werte.

Der Rausch-Generator stellt verschiedene Arten von Rauschen zur Verfügung, um Rauschen auf allen drei Ebenen abzubilden. Diese Arten sind an die in [11] (PLG) implementierten Arten des Rauschens angelehnt. Bei der Simulation kann für jede Art eine Wahrscheinlichkeit angegeben werden, die bestimmt, wie häufig diese Art des Rauschens nach dem Ausführen eines Knotens auftritt.

Doppelte Knotenausführung Diese Art des Rauschens sorgt dafür, dass ein Knoten doppelt ausgeführt wird [11].

Knoten überspringen Diese Art des Rauschens sorgt für das Überspringen eines oder mehrerer Knoten. Bei der Simulation kann die minimale und maximale Anzahl der zu überspringenden Knoten angegeben werden. Zufällig wird ein Wert aus diesem Intervall bestimmt und dieselbe Anzahl an Knoten übersprungen. [11].

Datenmanipulation Diese Art des Rauschens kann nur nach der Ausführung von Knoten auftreten, die einen Eintrag im Eventlog erzeugen. Hierbei werden die Attribute des erzeugten Ereignisses modifiziert. Aus allen Feldern im Eventlog (Aktivitäts-/Fall-Id, Zeitstempel und Attribute) wird zufällig eines ausgewählt und mit einem zufälligen Wert befüllt. Eine Ausnahme bildet der Zeitstempel, bei dem eine zufällige Zeitspanne addiert oder subtrahiert wird [11].

Sprung Rauschen Bei dieser Art des Rauschens wird ein zufälliger Knoten im Eventlog ausgewählt und als aktiver Knoten gesetzt. Dies führt zu einem „Sprung“ im Prozess, da dieser an einer anderen Stelle fortgeführt wird.

5.3.6 Attribute

Die Attribute der Eventlogs spielen eine zentrale Rolle in dem in Abschnitt 3 vorgestellten Konzept zur Identifikation von Fall- und Aktivitäts-Id. Es wird die Hypothese aufgestellt, dass Fall- und Aktivitäts-Id charakteristische Merkmale im Bezug auf die Wertwiederholung aufweisen, die sich von den anderen Attributen des Eventlogs unterscheiden. Daher ist es in der Prozess-Engine von großer Bedeutung, dass möglichst komplexe und praxisnahe Attribute abgebildet werden können. Die Prozess-Engine enthält daher unterschiedliche Arten von Attributen, die sich im Verhalten der Wertgenerierung unterscheiden.

Jedes Attribut besitzt einen Namen und kann mit einer Liste von Aktivitäten verknüpft werden. Die Beziehung zwischen Attributen und Aktivitäten ist dabei n zu n. Dies bedeutet, dass eine Aktivität mit mehreren Attributen und ein Attribut mit mehreren Aktivitäten verknüpft werden kann. Wird eine Aktivität ausgeführt, generieren alle verknüpften Attribute neue Werte. Dies wird im Folgenden als Aktivierung der Attribute bezeichnet. Die generierten Werte werden unter dem Namen der Attribute als Spalten in den von der Aktivität erzeugten Eventlog-Eintrag aufgenommen. Alle folgenden Eventlog-Einträge übernehmen diese Werte, auch wenn sie nicht mit den Attributen verknüpft sind. Somit bleibt der Wert eines Attributs konstant, bis das Attribute erneut durch eine Aktivität aktiviert wird.

Die Arten von Attributen unterscheiden sich im Verhalten der Wertgenerierung bei der Aktivierung. Diese Arten sind inspiriert von den Verhaltensweisen der Attribute aus realen Geschäftsprozessen:

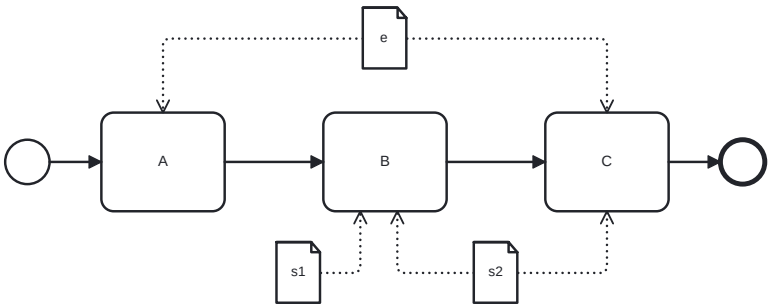
Statische Attribute Statische Attribute erzeugen immer denselben Wert. Dies erlaubt das Abbilden von booleschen Flags. Ein Beispiel hierfür ist ein Status, der angibt, ob eine Validierung durchgeführt wurde. Vor der Ausführung des Prüfschrittes ist der Wert des statischen Attributs leer. Nach der Durchführung der Validierung enthält das Attribut einen festen Wert, der bis zum Ende des Prozesses konstant bleibt.

Eindeutige Attribute Eindeutige Attribute erzeugen immer einen neuen, einzigartigen Wert bei der Aktivierung. Mit diesen Attributen kann beispielsweise das Verhalten von Session-Ids, Transaktions-Ids oder Zeitstempeln abgebildet werden.

Listen-Attribute Listen-Attribute enthalten eine Liste von Werten. Bei der Ausführung einer verknüpften Aktivität wird zufällig ein Wert aus dieser Liste ausgewählt und zurückgegeben. Es ist möglich, dass derselbe Wert mehrfach ausgewählt wird. Diese Attributart ermöglicht es, Szenarien abzubilden, bei denen dem Prozess nur eine begrenzte Anzahl an Ressourcen zur Verfügung steht. Beispielsweise können mehrere Systeme existieren, die eine Aktivität ausführen können, oder es gibt nur eine begrenzte Anzahl an Benutzern, die berechtigt sind, bestimme Aktionen zu tätigen.

Sequenzielle Attribute Diese Attribute enthalten ebenfalls eine Liste von Werten, die jedoch sequenziell abgearbeitet werden. Bei jeder Aktivierung wird der nächste Wert der Liste zurückgegeben. Ist das Ende der Liste erreicht, wird für alle weiteren Aufrufe der letzte Wert verwendet. Mit diesen Attributen können Status-Flags simuliert werden, die typischerweise feste Werte durchlaufen. Ein Beispiel hierfür ist der Bestellstatus, bei dem die Werte *eingegangen*, *in Bearbeitung*, *Versand*, und *zugestellt* in einer festen Reihenfolge vorkommen.

Zufällig sequenzielle Attribute Diese Attributart ähnelt den sequenziellen Attributen, unterscheidet sich jedoch dadurch, dass die Liste der Werte bei der ersten Aktivierung zufällig sortiert wird. Dadurch entsteht in verschiedenen Prozessinstanzen eine unterschiedliche Reihenfolge der Werte. Diese Art der Attribute ermöglicht die Simulation von Prozessen, bei denen die Reihenfolge von Ereignissen variieren kann. Ein Beispiel hierfür sind unterschiedliche Prüfungsreihenfolgen in einer Qualitätssicherung, bei denen die Reihenfolge der Prüfschritte von Instanz zu Instanz unterschiedlich ist.



Fall-Id	Aktivitäts-Id	s1	s2	e	Timestamp
0	A			a3cd8260	00:00
0	B	33af1182	2f4a2c35	a3cd8260	00:01
0	C	33af1182	2f4a2c35	f8e1a11f	00:02
1	A			ee12fc48	00:03
1	B	33af1182	2f4a2c35	ee12fc48	00:04
1	C	33af1182	2f4a2c35	b71d0adb	00:05

Abbildung 11: Beispiel für die Wertgenerierung mit den statischen Attributen $s1$, $s2$ und dem eindeutigen Attribut e

Abbildung 11 zeigt beispielhaft die Wertgeneration der statischen und eindeutigen Attribute. Im Prozess enthalten ist das statische Attribut $s1$, dass mit Aktivität B verknüpft ist, das statische Attribut $s2$, dass mit Aktivität B und C verknüpft ist und das eindeutige Attribut e, dass mit den Aktivitäten A und C verknüpft ist. Im unteren Bereich der Abbildung ist der zugehörige Eventlog mit zwei Prozessinstanzen abgebildet.

Dem Eventlog kann entnommen werden, dass für die Aktivitäts-Id A keine Werte für $s1$ und $s2$ existieren. Erst bei der Ausführung der verknüpften Aktivität B erhalten diese Werte. Da $s1$ und $s2$ unterschiedliche statische Attribute sind, erstellen sie bei der Aktivierung verschiedene Werte. Allerdings erzeugen sie dieselben Werte in allen Prozessinstanzen. Ein erneuter Aufruf von $s2$ in der Aktivität C führt zu keiner Wertänderung, da ein statisches Attribut immer den gleichen Wert zurück gibt. Das eindeutige Attribut e erzeugt bei jeder Aktivierung durch die Aktivitäten A und C einen neuen eindeutigen Wert. Implementiert ist dies über eine UUID, um sicherzustellen, dass die Werte eindeutig sind. Zur Übersicht werden in der Abbildung nur die ersten Zeichen der UUIDs abgebildet. Die Werte für $s2$ im Eventlog-Eintrag von Akivität B sind identisch mit denen von Aktivität A, da $s2$ nicht mit B verknüpft ist und somit nicht aktiviert wird. Auch ist zu sehen, dass $s2$ eindeutige Werte instanzübergreifend erzeugt.

5.4 Der Prozessgenerator

In diesem Unterabschnitt wird beschrieben, wie zufällige Geschäftsprozesse mit dem Prozessgenerator erzeugt werden. Dieser baut auf der Prozess-Engine auf und verwendet diese für die Erzeugung und Simulation der Prozesse. Fokus bei der Implementierung liegt auf der Erfüllung der in Unterabschnitt 5.1 beschriebenen Anforderungen. Die Generierung der Prozesse kann in 6 Schritte unterteilt werden:

1. **Generierung der Entitäten:** Alle Entitäten des Prozesses werden gemäß den Parametern der Konfigurationsdatei generiert. Entitäten sind alle Elemente, die in einem Prozess vorkommen, wie Aktivitäten, Entscheidungen oder Attribute.
2. **Generierung der Subpfade:** Die Aktivitäten werden aneinandergereiht, was zur Entstehung der Subpfade des Prozesses führt.
3. **Zusammensetzen der Subpfade:** Die Subpfade werden mit den Entscheidungen verknüpft und zu dem finalen Prozess zusammengesetzt.
4. **Attribute verknüpfen:** Die Attribute werden mit den Aktivitäten des Prozesses verknüpft.
5. **Sprungpfade referenzieren:** Die Referenzen der Sprungpfade werden auf zufällige Knoten im Geschäftsprozess gesetzt.
6. **Prozess validieren:** Es wird geprüft ob der Prozess in sich korrekt und valide ist.

5.4.1 Die Konfigurationsdatei

Die Generierung aller Entitäten wird über eine Konfigurationsdatei gesteuert. Parameter wie die Anzahl der Aktivitäten, die Anzahl der Entscheidungen sowie die Arten und Häufigkeit der Attribute können in dieser Datei angegeben werden. Viele der Parameter definieren ein Intervall, aus dem bei der Erzeugung der Entität ein zufälliger Wert generiert wird. Ist beispielsweise das Intervall für die Anzahl der Aktivitäten auf $[1, 10]$ gesetzt, werden Prozesse mit 1 bis 10 Aktivitäten generiert. Dies ermöglicht die Erzeugung variierender Prozesse mit derselben Konfiguration.

```
{
  ...
  "activities": {
    "count": [5, 10],
    "duration": [10, 20],
    "deviation": [1, 10]
  },
  ...
}
```

Auflistung 2: Abschnitt aus der Konfigurationsdatei für Aktivitäten.

5.4.2 Generierung der Aktivitäten

Ein Auszug aus der Konfigurationsdatei für die Aktivitätsgenerierung ist in Auflistung 2 abgebildet. Die Generierung der Aktivitäten erfolgt anhand der Parameter *count*, *duration* und *deviation*.

Jeder dieser Parameter beschreibt einen Wertebereich, aus dem jeweils ein zufälliger Wert generiert wird. Diese generierten Werte werden im Folgenden als *count′*, *duration′* und *deviation′* bezeichnet.

Es werden insgesamt *count′* einzigartige Aktivitäten generiert. Diese Aktivitäten erhalten Namen, beginnend mit dem Buchstaben A, gefolgt von einer eindeutigen, fortlaufenden Nummerierung.

Für jede Aktivität werden individuelle Werte für *duration′* und *deviation′* erzeugt. Die Variable *duration′* repräsentiert die durchschnittliche Ausführungsdauer der Aktivität. Während der Ausführung kann die tatsächliche Durchführungszeit einer Aktivität von dieser durchschnittlichen Dauer abweichen. Die Größenordnung dieser Abweichung wird durch die Standardabweichung *deviation′* gesteuert.

5.4.3 Generierung der Bedingungen

Eine beispielhafte Konfiguration für die Generation der Bedingungen ist in Auflistung 3 zu sehen. Der Generator für die Bedingungen kann über die Parameter *count*, *branches* und *percentages* gesteuert werden, die alle einen Wertebereich angeben. Der Parameter *crossBranch* gibt die Wahrscheinlichkeit an, mit der ein Pfad zu einem Sprungpfad wird.

```
{
  ...
  "conditions": {
    "count": [2, 3],
    "branches": [2, 4],
    "percentage": [0.05, 0.08],
    "crossBranch": 0.10
  }
  ...
}
```

Auflistung 3: Abschnitt aus der Konfigurationsdatei für Bedingungen.

Zunächst wird ein zufälliger Wert aus dem Intervall *count* erzeugt und entsprechend viele Bedingungen werden generiert. Für jede Bedingung wird ein zufälliger Wert im Intervall von *branches* erzeugt und die gleiche Anzahl an Ausgangspfaden wird an die Bedingung angehängt. Gemäß der Wahrscheinlichkeit aus *crossBranch* besteht für jeden Ausgangspfad die Chance, dass dieser als Sprungpfad markiert wird. Am Ende der markierten Pfade wird nach der Generation des Prozesses ein Sprungknoten angehängt, der auf einen beliebigen Knoten im Prozess verweist (siehe Unterunterabschnitt 5.4.7). Jeder Zweig erhält eine zufällige Wahrscheinlichkeit, die im Wertebereich von *percentages* liegt.

Die Ermittlung dieser Wahrscheinlichkeiten unter Berücksichtigung der Intervallgrenzen ist keine triviale Aufgabe, da auf die Unvoreingenommenheit der erzeugten Werte Rücksicht genommen werden muss. Eine Voreingenommenheit in den Wahrscheinlichkeiten einer Bedingung hat Einfluss auf die Auftrittswahrscheinlichkeit der entstehenden Muster und beeinflusst damit die errechneten Kennzahlen. Es besteht die Möglichkeit, dass das intelligente System im Trainingsprozess diese Voreingenommenheit erlernt und nicht die Charakteristiken der Fall- und Aktivitäts-Id (siehe Absatz 2.3.1).

Für die Generierung der Wahrscheinlichkeiten wird daher die Dirichlet-Verteilung verwendet. Diese gewährleistet eine unvoreingenommene Verteilung der Werte. Algorithmus 5 beschreibt die Erzeugung der Wahrscheinlichkeiten unter Berücksichtigung der Intervallgrenzen *percentages* aus der Konfigurationsdatei mithilfe der Dirichlet-Verteilung.

Zunächst wird der *alpha*-Vektor in der Größe der zu erstellenden Wahrscheinlichkeiten erzeugt und alle Werte werden mit 1 befüllt. Dies sorgt dafür, dass alle generierten Wahrscheinlichkeiten gleich gewichtet und somit gleichverteilt sind. Die Funktion, welche

Algorithmus 5 Erzeugung der Wahrscheinlichkeiten einer Bedingung

```
percentages := Wertebereich der Wahrscheinlichkeiten aus der Konfigurationsdatei
branches := Anzahl der zu erstellenden Pfade
isFinished = false
alpha = array(branches).fill(1)
while !isFinished do
    isFinished = true
    p = dirichlet(branches, alpha)
    for all b ∈ branches do
        if p[b] < percentages[0] and p[b] > percentages[1] then
            isFinished = false
        end if
    end for
end while
return p
```

die Wahrscheinlichkeiten nach der Dirichlet-Verteilung berechnet, wird daraufhin ausgeführt und die Ergebnisse werden in p gespeichert. Für alle Werte von p wird geprüft, ob die Intervallgrenzen von *percentages* eingehalten werden. Ist dies nicht der Fall, werden neue Wahrscheinlichkeiten für p berechnet. Andernfalls wird p als finales Ergebnis angesehen.

In Java existieren nur wenige Bibliotheken, die die Dirichlet-Verteilung implementieren [20]. Diese Arbeit verwendet die private Bibliothek *Java Statistical Analysis Tool* (JSAT) für die Berechnung der Verteilung [20]. Um die Korrektheit der Implementierung und des Algorithmus 5 zu gewährleisten, wird ein Versuch durchgeführt, der die Verteilung der Wahrscheinlichkeiten untersucht.

Durchgeführt wird ein 2^k-Versuch, der die Verteilung mit und ohne Berücksichtigung der Intervallgrenzen untersucht. Bedingungen mit 3 Ausgangspfaden werden erstellt, da dies die größte Dimension ist, die sich visuell als Plot darstellen lässt. Werden die Wahrscheinlichkeiten in einem Streudiagramm dargestellt, bilden sie eine Ebene im dreidimensionalen Raum, die auf Hotspots untersucht werden kann. Für jede der zu testenden Intervallgrenzen $[0, 1], [0.2, 1], [0, 0.8], [0.1, 0.4]$ werden 100 000 Bedingungen erstellt und als Streudiagramm dargestellt.

In Abbildung 12 werden die Ergebnisse des Versuchs dargestellt. Die Achsen geben die Wahrscheinlichkeit der Ausgangspfade in Prozent an. Je nach Intervallgrenze ist der Wertebereich der Punktebene größer oder kleiner. Wie erwartet sind die Werte alle homogen

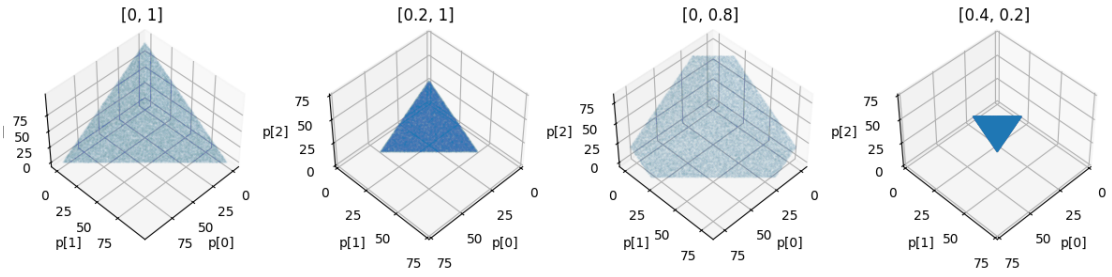


Abbildung 12: Erzeugte Verteilung der Wahrscheinlichkeiten aus dem Algorithmus 5 für verschiedene Intervallgrenzen

verteilt und es entstehen keine Hotspots innerhalb der Punktebene. Auch bei Verwendung der Intervallgrenzen bleibt dieses Verhalten bestehen.

Mit dem Versuch wird somit belegt, dass die Wahrscheinlichkeiten gleichverteilt und damit unvoreingenommen sind.

5.4.4 Generierung der Attribute

```
{
  ...
  "attributes": [{
    "enablePercentage": 0.5,
    "variant": "RANGE",
    "count": [1, 2],
    "attachments": [2, 4],
    "numValues": [2, 4]
  }]
  ...
}
```

Auflistung 4: Abschnitt aus der Konfigurationsdatei für Attribute.

In Auflistung 4 ist beispielhaft die Konfiguration der Attribute aus der Konfigurationsdatei zu sehen. Für eine Konfigurationsdatei ist es möglich mehrere Attributskonfigurationen zu hinterlegen. Im Folgenden sind die Parameter der Konfiguration erläutert:

- **EnablePercentage:** Dieser Parameter gibt die Wahrscheinlichkeit an, mit der diese Attributkonfiguration aktiv ist. Nur aus aktiven Konfigurationen werden Attribute für den Geschäftsprozess generiert. Dies diversifiziert die erzeugten Ge-

schäftsprozesse, da bei der Generierung mehrerer Prozesse aus derselben Konfiguration nicht immer dieselben Attribute enthalten sind.

- **Variant:** Der Parameter *variant* bestimmt die Art der Attribute. In Unterunterabschnitt 5.3.6 sind alle Attributarten beschrieben.
- **Count:** Dieser Parameter gibt an, wie viele Attribute aus der Attributkonfiguration generiert werden. Somit ist es möglich, aus derselben Attributkonfiguration mehrere, unabhängige Attribute zu erzeugen.
- **Attachments:** Dieser Parameter bestimmt, mit wie vielen Aktivitäten ein aus dieser Konfiguration generiertes Attribut verknüpft wird. Jedes Attribut erhält einen eigenen zufälligen Wert im Intervall.
- **NumValues:** Dieser Parameter wird nur für Listenattribute sowie für sequenzielle und zufällig sequenzielle Attribute verwendet. Diese Attribute besitzen eine Liste an möglichen Werten, die sie annehmen können. Mit dem Parameter kann die Länge der Wertliste bestimmt werden. Für jedes erstellte Attribut wird ein zufälliger Wert aus dem Intervall generiert.

5.4.5 Generierung der Subpfade

Sind alle Entitäten erzeugt, beginnt die Erstellung der Subpfade des Prozesses. Ein Subpfad ist ein Pfad, der an einer Bedingung startet und endet, wenn alle Pfade dieser Bedingung wieder vereinigt werden. Zusätzlich gibt es einen Subpfad, der am Startknoten beginnt und am Endknoten endet.

Ein Knoten im Prozess kann immer nur einem Subpfad zugewiesen werden. Das bedeutet, wenn sich auf einen Subpfad s eine Bedingung befindet, die die Subpfade $s_1, ..., s_n$ erzeugt, werden die Knoten, die in $s_1, ..., s_n$ enthalten sind, nicht als Bestandteil von s angesehen.

Für die Generierung der Subpfade wird zunächst die Anzahl der im Prozess enthaltenen Subpfade bestimmt. Diese ergibt sich aus der Summe aller Ausgangspfade der Bedingungen plus dem Subpfad, der vom Start- zum Endknoten führt. In einem Geschäftsprozess, der zwei Bedingungen mit jeweils 3 Ausgangspfaden enthält, existieren somit 7 Subpfade.

Auf diese Subpfade werden die Aktivitäten verteilt, wobei besondere Rücksicht auf die Unvoreingenommenheit der Verteilung gelegt wird. Die Subpfade sollen möglichst eine hohe Variation der Pfadlängen aufweisen. Dies erhöht die Unvoreingenommenheit der Geschäftsprozesse, da keine Präferenz gegenüber bestimmten Pfadlängen entsteht.

Algorithmus 6 Verteilung der Aktivitäten auf die Subpfade

```
activities := Liste der Aktivitäten
conditions := Liste der Bedingungen
entities = array(activities, conditions).shuffle()
activeBranches := array(1)
for all e ∈ entities do
    idx = random(branches.length)
    activeBranches[idx].add(e)
    if e ∈ activities then
        idx = random(activeBranches.length())
        activeBranches[idx].add(a)
    else if e ∈ conditions then
        e.getBranches().forEach(branch → activeBranches.add([]))
    end if
end for
```

Algorithmus 6 zeigt den Pseudocode für die Verteilung der Aktivitäten auf die Subpfade. Zu Beginn werden alle Bedingungen (*conditions*) und Aktivitäten (*activities*) in die Liste *entities* aufgenommen und in zufälliger Reihenfolge sortiert. Die Liste *activeBranches* enthält alle aktiven Subpfade, wobei zunächst nur der leere Hauptpfad enthalten ist. Im nächsten Schritt wird iterativ jede Entität e aus *entities* verarbeitet. Falls e eine Aktivität ist, wird diese einem zufälligen Subpfad p aus *activeBranches* zugeordnet. Ist e hingegen eine Bedingung, werden alle Ausgangspfade der Bedingung als leere Subpfade zu *activeBranches* hinzugefügt. Auf diese Weise werden nacheinander alle Subpfade des Geschäftsprozesses zu *activeBranches* hinzugefügt und mit Aktivitäten befüllt.

Die Pfadlänge ist in diesem Algorithmus von der Position der Bedingungen in der *entities*-Liste abhängig. Wenn eine Bedingung durch die zufällige Sortierung im vorderen Segment der Liste steht, ist zu erwarten, dass ihre Subpfade viele Aktivitäten aufweisen, da sie öfter die Möglichkeit haben, Aktivitäten zu erhalten. Die Subpfade einer Bedingung, die sich hinten in der Liste befinden, werden in der Regel wenige Aktivitäten aufweisen, da

die Pfade erst zu einem späteren Zeitpunkt hinzugefügt werden. Da die *entities*-Liste zufällig sortiert ist, wird angenommen, dass die Verteilung der Pfadlängen ebenfalls zufällig und damit unvoreingenommen ist.

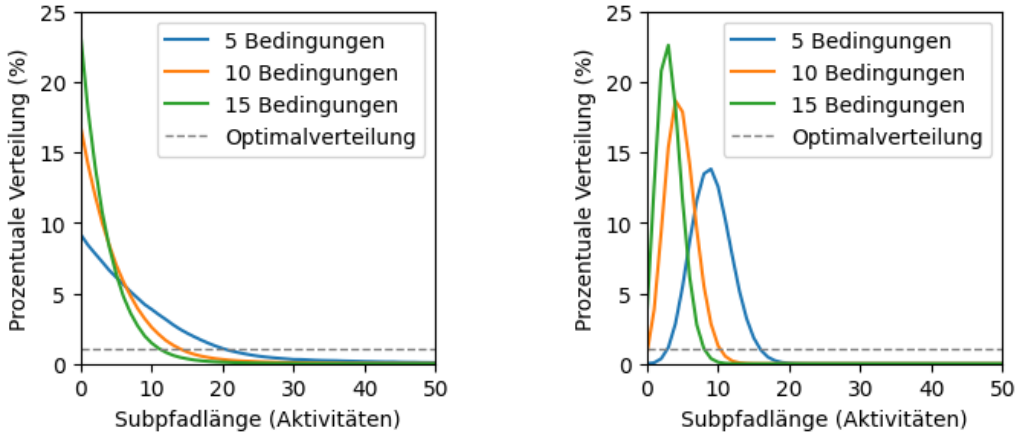
Algorithmus 7 Erzeugung der Wahrscheinlichkeiten einer Bedingung

```
activities := Liste der Aktivitäten
numBranches := Anzahl an Pfaden im Prozess
branches = array(numBranches)
for all a ∈ activities do
    idx = random(numBranches)
    branches[idx].add(a)
end for
```

Um diese Annahme zu belegen, wird der oben beschriebene Algorithmus mit einer einfachen Referenzimplementierung gegenübergestellt. Der Pseudocode der Referenzimplementierung wird in Algorithmus 7 dargestellt. Zu Beginn wird eine leere Liste für alle Subpfade im Prozess angelegt. Für jede Aktivität a wird ein zufälliger Subpfad ausgewählt und diese dem Ende des Pfads angehängt. Es ist zu erwarten, dass die Verteilung der Pfadlängen der Referenzimplementierung einer Normalverteilung ähnelt, da die Aktivitäten zwar zufällig, aber bei vielen Aktivitäten dennoch gleichmäßig auf die Subpfade verteilt werden. Somit entsteht eine erwartete Pfadlänge.

Die tatsächliche Verteilung der Pfadlängen des Algorithmus 6 und der Referenzimplementierung wird durch ein Versuch untersucht. Es werden für beide Ansätze jeweils 100 000 Geschäftsprozesse mit je 100 Aktivitäten für die Szenarien mit 5, 10 und 15 Bedingungen im Prozess erstellt. Jede Bedingung verfügt über zwei Ausgangspfade. In den generierten Geschäftsprozessen wird die Länge der Subpfade und die Länge der Gesamtpfade analysiert.

Länge der Subpfade In Abbildung 13 wird die prozentuale Verteilung der Subpfadlängen dargestellt. Die graue gestrichelte Linie repräsentiert die optimale Verteilung bei vollkommener Unvoreingenommenheit. Bei dieser besitzen alle Pfadlängen die gleiche Auftrittswahrscheinlichkeit. Da Prozesse mit 100 Aktivitäten erzeugt werden, existieren 100 mögliche Pfadlängen, weshalb diese Linie auf der Höhe 1 Prozent liegt. Je weiter die restlichen Kurven von diesem Optimum abweichen, desto stärker ist die Voreingenommenheit der Pfadlänge. Aus dem Abbildung 13a kann direkt abgelesen werden, dass der Algorithmus stets einen größeren Wertebereich der Pfadlängen abdeckt als die Referenzimplementierung. Beispielsweise umfasst der Algorithmus 6 bei 15 Bedingungen noch



(a) Verteilung der Subpfadlängen des Algorithmus 6

(b) Verteilung der Subpfadlängen der Referenzimplementierung

Abbildung 13: Prozentuale Verteilung der Subpfadlängen

Pfadlängen im Wertebereich $[0, 60]$, während die Referenzimplementierung nur das Intervall $[0, 15]$ abdeckt. Auch sind die Höhepunkte des Algorithmus bis zu 15 Bedingungen niedriger als die der Referenz.

Es wird die *mittlere quadratische Abweichung* (engl. *Mean Suare Error, MSE*) verwendet, um die Abweichungen der Pfadlängen von der Optimalverteilung zu berechnen. Diese eignet sich besonders als Fehlermaß, da Ausreißer stärker gewichtet werden. Werte, die stark von dem Optimum abweichen, sorgen für eine hohe Voreingenommenheit. Daher ist es vorteilhaft, wenn diese stärker in das Fehlermaß einfließen.

Die MSE wird für jede Kurve mit der folgenden Formel berechnet:

$$\frac{1}{100} \cdot \sum_{i=0}^{100} (f(i) - 0.01)^2$$

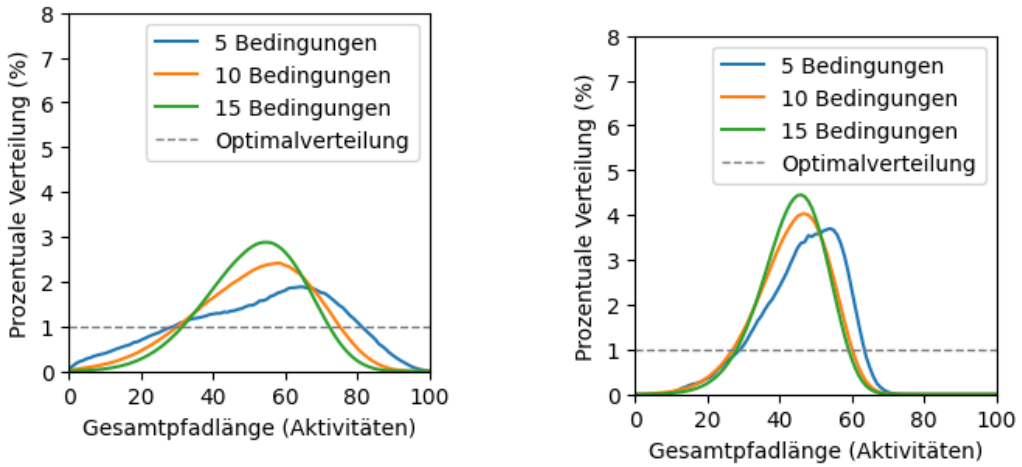
Für jede Pfadlänge i wird die prozentuale Verteilung der Pfadlänge $f(i)$ ermittelt. Die Abweichung einer Pfadlänge ergibt sich aus der Differenz zwischen der prozentualen Verteilung und dem Optimum (1 Prozent). Dieser Wert wird quadriert und für alle Pfadlängen aufsummiert. Die MSE ergibt sich aus dem Durchschnitt dieser Summe über alle 100 möglichen Pfadlängen.

In Tabelle 4 sind alle errechneten Abweichungen des Algorithmus 6 und der Referenzimplementierung enthalten. Auffällig ist, dass die Abweichung bei beiden Ansätzen mit

	MSE bei 5 Bedingungen	MSE bei 10 Bedingungen	MSE bei 15 Bedingungen
Algorithmus 6	4.148685	8.399618	12.277246
Referenzimplementierung	8.774330	12.277672	15.145451
Differenz	4.625644	3.878055	2.868205

Tabelle 4: Mittlere quadratische Abweichung der Subpfadlängen des Algorithmus 6 und der Referenzimplementierung

steigender Anzahl an Bedingungen zunimmt. Dies ist damit zu begründen, dass die Aktivitäten auf mehr Pfade aufgeteilt werden und somit kürzere Pfade mit höherer Wahrscheinlichkeit entstehen. Pfade mit vielen Aktivitäten sind somit selten und weiter von dem Optimum entfernt. Die Abweichung des Algorithmus ist stets niedriger als die der Referenzimplementierung. Dies ist besonders stark ausgeprägt, bei Prozessen mit wenigen Bedingungen. Mit steigender Anzahl an Bedingungen gleichen sich die Abweichungen des Algorithmus und der Referenz zunehmend an.



(a) Verteilung der Gesamtpfadlängen des Algorithmus 6

(b) Verteilung der Gesamtpfadlängen der Referenzimplementierung

Abbildung 14: Prozentuale Verteilung der Gesamtpfadlängen

Länge der Gesamtpfade Ein Gesamtpfad ist ein Pfad, der am Startknoten beginnt und am Endknoten endet. Er kann als die Summe der Subpfade betrachtet werden, die durchlaufen werden, um vom Startknoten den Endknoten zu erreichen. Anders als bei den Subpfaden werden alle Knoten, die auf dem Pfad durchlaufen werden, als Bestandteil des Gesamtpfades angesehen. Eine Aktivität bzw. ein Subpfad kann somit in mehreren

Gesamtpfaden enthalten sein.

In Abbildung 14 wird die Verteilung der Gesamtpfadlängen des Algorithmus 6 (Abbildung 14a) mit der Referenzimplementierung (Abbildung 14b) gegenübergestellt. Es ist direkt einsehbar, dass die Extremwerte der Referenzimplementierung deutlich höher ausfallen, als die des Algorithmus. Zudem deckt der Algorithmus für alle Bedingungen einen größeren Wertebereich der möglichen Pfadlängen ab, während der Wertebereich der Referenzimplementierung beschränkt ist. Für 15 Bedingungen enthält der Algorithmus Pfadlängen im Intervall $[0, 98]$, währenddessen deckt die Referenz nur das Intervall $[0, 78]$ ab.

	MSE bei 5 Bedingungen	MSE bei 10 Bedingungen	MSE bei 15 Bedingungen
Algorithmus 6	0.330327	0.720383	1.056756
Referenzimplementierung	1.652513	1.838649	2.131881
Differenz	1.322187	1.118265	1.075125

Tabelle 5

Für alle Graphen werden ebenfalls die MSE berechnet, um die Abweichung von der Optimalverteilung zu erhalten. Diese weisen dasselbe Verhalten wie die Subpfade auf. Die Abweichung des Algorithmus ist stets geringer als die der Referenzimplementierung. Mit zunehmender Anzahl an Bedingungen, steigt die Abweichung für beide Ansätze. Ebenfalls steigt die Abweichung mit zunehmender Anzahl der Bedingungen an. Die Differenz zwischen dem Algorithmus und der Referenzimplementierung sinkt mit zunehmender Anzahl an Bedingungen.

Fazit Sowohl aus der Subpfadlänge als auch aus der Gesamtpfadlänge geht hervor, dass eine leichte Voreingenommenheit gegenüber bestimmten Pfadlängen existiert. Der vorgestellte Algorithmus liefert jedoch in allen untersuchten Szenarien eine kleinere Abweichung zur Optimalverteilung als die Referenzimplementierung. Trotz der Voreingenommenheit können durch den Algorithmus nahezu alle möglichen Gesamtpfadlängen in den Prozessen auftreten. Aus diesen Gründen wird der Algorithmus 6 für die Erstellung der Subpfade verwendet.

5.4.6 Zusammensetzen der Pfade

Die Subpfade des vorherigen Schrittes können nun mit den Bedingungen verbunden werden, sodass der finale Geschäftsprozess entsteht.

Algorithmus 8 Erzeugung der Wahrscheinlichkeiten einer Bedingung

```
subpaths := Liste der Subpfade, die die Aktivitäten enthalten
conditions := Liste aller Bedingungen
entrypoints = array()

mainPath = subpaths.get(random(subpath.length()))
subpaths.remove(mainPath)
mainPath.getActivities().forEach(a → entrypoints.add(a)
for all c ∈ conditions do
    entrypoint = entrypoints.get(random(entrypoints.length()))
    c.setNext(entrypoint.getNext())
    entrypoint.setNext(c)
    entrypoints.add(c)

    for all b ∈ c.getBranches() do
        subpath = subpaths.get(random(subpath.length()))
        subpaths.remove(subpath)
        b.setNext(subpath.get(0))
        subPath.getActivities().forEach(a → entrypoints.add(a))
    end for
end for
```

Der Pseudocode für das Zusammensetzen der Subpfade wird in Algorithmus 8 dargestellt. Für den Pseudocode wird angenommen, dass die Aktivitäten der Subpfade (*subpaths*) bereits aufeinander verweisen. Dies bedeutet, dass die *getNext*-Methode einer Aktivität auf den nächsten Knoten im Subpfad referenziert.

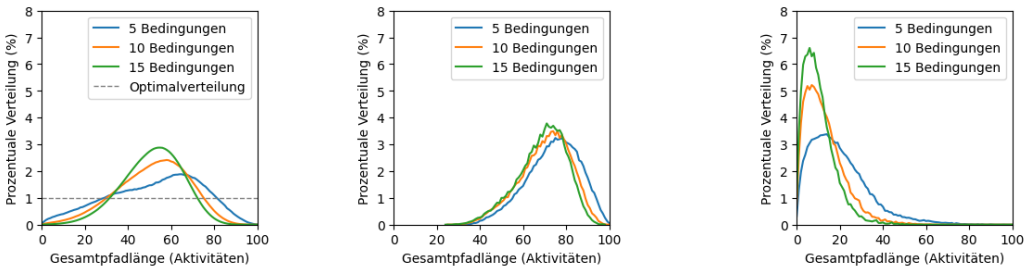
Aus allen Subpfaden wird ein zufälliger Pfad ausgewählt und aus der Liste entfernt. Dieser Pfad wird als Hauptpfad (*mainPath*) angesehen, der den Start- und Endknoten verbindet. Alle Knoten dieses Pfades werden in die Liste der möglichen Einstiegspunkte (*entrypoints*) aufgenommen. Neue Subpfade können nur an den Einstiegspunkten dem Prozess hinzugefügt werden.

Für jede Bedingung (c) aus allen Bedingungen (*conditions*) werden folgende Schritte ausgeführt:

- Es wird ein zufälliger Einstiegspunkt ausgewählt und die Bedingung hinter diesem angehängt.
- Jeder Ausgang der Bedingung wird mit einem Subpfad verbunden. Die verbundenen Subpfade werden aus *subpaths* entfernt. Alle Aktivitäten der verbundenen Subpfade werden in die Liste der Einstiegspunkte aufgenommen.
- Die Bedingung wird in die Liste der Einstiegspunkte aufgenommen.

Sind alle Bedingungen durchlaufen, ist die Generierung des Prozesses abgeschlossen. Dieser ist nun in *mainBranch* enthalten.

Im vorherigen Unterunterabschnitt wird bereits auf die Voreingenommenheit der Gesamtpfadlängen eingegangen, um den Algorithmus für die Erzeugung der Subpfade zu evaluieren. Im Folgenden werden weitere Analysen zur Bestimmung der Unvoreingenommenheit der Pfadlängen durchgeführt. Es werden dieselben Prozesse des vorherigen Versuchs verwendet.



(a) Verteilung aller Gesamtpfade (b) Verteilung der längsten Pfade jedes Prozesses (c) Verteilung der kürzesten Pfade jedes Prozesses

Abbildung 15: Prozentuale Verteilung der Gesamtpfadlängen

In Abbildung 15a wird die Verteilung der Gesamtpfadlängen dargestellt. Dieser kann entnommen werden, dass eine Voreingenommenheit vorliegt, da die Kurven von der Optimalverteilung (graue Linie) abweichen. Die maximale Abweichung tritt bei 15 Bedingungen auf und beträgt 1.87 Prozent (Höhepunkte der grünen Linie). In den Szenarien mit 10 und 15 Bedingungen sind Pfadlängen im Bereich von [0, 100] enthalten, wodurch der gesamte Wertebereich der möglichen Pfadlängen abgedeckt wird. Lediglich bei 15 Bedingungen sind nur Pfadlängen im Wertebereich [0, 98] vorhanden.

Zusätzlich wird die Verteilung nur des längsten Pfades (Abbildung 15b) bzw. des kürzesten Pfades (Abbildung 15c) der Prozesse isoliert betrachtet. Aus Abbildung 15b geht hervor, dass die Anzahl der Bedingungen nur einen geringen Einfluss auf den längsten Pfad der Prozesse hat. Bei 5 Bedingungen liegt der Höhepunkt bei 3.24 Prozent und einer Pfadlänge von 75 Aktivitäten. Dies unterscheidet sich nur geringfügig von 15 Bedingungen, bei denen der Höhepunkt bei 3.78 Prozent und einer Pfadlänge von 71 Aktivitäten liegt.

Das Verhalten der kürzesten Pfade ist dazu gegensätzlich. Die Wahrscheinlichkeit für sehr kurze Pfade steigt mit zunehmender Anzahl an Bedingungen stark an. Bei 5 Bedingungen beträgt die am häufigsten vertretene Pfadlänge noch 14 Aktivitäten, während diese bei 15 Bedingungen lediglich 6 Aktivitäten lang ist. Dieses Verhalten lässt sich durch das Verfahren zur Subpfaderstellung gemäß Algorithmus 6 erklären. Dieses Verfahren erzeugt bei einer größeren Anzahl an Aktivitäten viele Subpfade mit geringer Pfadlänge (siehe Abbildung 13a).

Bedingungen	**kürzester Pfad < u**			**Längster Pfad > o**		
	$u = 30$	$u = 20$	$u = 10$	$o = 70$	$o = 80$	$o = 90$
5	82.02 %	59.48 %	27.42 %	66.12 %	35.33 %	8.91 %
10	95.51 %	82.51 %	47.11 %	56.27 %	22.99 %	2.67 %
15	98.30 %	90.95 %	58.96 %	52.44 %	17.10 %	1.03 %

Tabelle 6: Wahrscheinlichkeit der Pfadlänge des kürzesten/längsten Pfades

Werden erneut alle Pfade der Geschäftsprozesse betrachtet (Abbildung 15a), tritt Die höchste Voreingenommenheit bei 15 Bedingungen auf. Das 90-Prozent-Konfidenzintervall um den Höhepunkt liegt in diesem Szenario bei $[30, 73]$ Aktivitäten. Dies bedeutet, dass nur 5 Prozent der Pfadlängen kleiner als 30 und 5 Prozent der Pfadlängen größer als 73 Aktivitäten sind. Mithilfe der kürzesten und längsten Pfadlängen wird berechnet, wie wahrscheinlich es ist, dass diese Pfade unter bzw. über bestimmten Pfadlängen liegen. Die Ergebnisse werden in Tabelle 6 dargestellt. Daraus geht hervor, dass trotz der Unvoreingenommenheit eine hohe Wahrscheinlichkeit besteht, dass in den generierten Prozessen mindestens ein sehr langer oder sehr kurzer Pfad existiert. Obwohl bei 15 Bedingungen die Wahrscheinlichkeit für Pfade mit mehr als 73 Aktivitäten unter 5 Prozent liegt, besteht eine Chance von 17.1 Prozent, dass ein Prozess mindestens einen Pfad mit mehr als 80 Aktivitäten enthält. Die Wahrscheinlichkeit, dass ein Pfad kürzer als 20 Aktivitäten ist, liegt bei 90.95 Prozent und ist damit noch höher.

Fazit In den erzeugten Prozessen zeigt sich eine Voreingenommenheit gegenüber bestimmten Pfadlängen. Dennoch weist die Verteilung der generierten Pfadlängen nur eine geringe Abweichung von der optimalen Verteilung auf. Zudem hat die Anzahl der Bedingungen einen Einfluss auf die Voreingenommenheit. Prozesse mit wenigen Bedingungen sind weniger voreingenommen als solche mit vielen Bedingungen. Es wird auch gezeigt, dass trotz dieser Voreingenommenheit nahezu alle möglichen Pfadlängen in den erzeugten Prozessen auftreten. Obwohl die Wahrscheinlichkeit für sehr kurze bzw. sehr lange Pfade generell gering ist, besteht eine hohe Wahrscheinlichkeit, dass Prozesse mindestens einen sehr langen bzw. sehr kurzen Pfad enthalten. Aus diesen Gründen wird die Voreingenommenheit der Pfadlängen als gering eingestuft.

5.4.7 Sprungpfade referenzieren

Bei der Generierung der Bedingungen werden bestimmte Pfade als Sprungpfade markiert (siehe Unterunterabschnitt 5.4.3). Damit diese zu Sprungpfaden werden, müssen Sprungknoten an deren Ende angehängt werden. Jeder als Sprungpfad markierte Pfad wird durchlaufen. Ein zufälliger Knoten im Prozess, der nicht auf dem Sprungpfad liegt, wird ausgewählt. Wird ein Knoten selektiert, der auf dem Sprungpfad liegt, entsteht ein Zyklus, der nicht durchbrochen werden kann. An das Ende des Sprungpfades wird ein Sprungknoten angehängt, der auf den zufällig ausgewählten Knoten verweist.

Sprungpfade sorgen in der Theorie für „Abkürzungen“ und Zyklen im Geschäftsprozess. In einem Versuch wird ermittelt, wie sich unterschiedliche Sprungpfadwahrscheinlichkeiten auf die Pfadlängen auswirken. Hierfür werden jeweils 100 000 Geschäftsprozesse mit 100 Aktivitäten, 10 Bedingungen und einer Sprungpfadchance von 0, 30, 50 und 70 Prozent generiert. Jede Bedingung spaltet den Eingangspfad in 3 Subpfade auf. Im Vergleich zu den vorherigen Versuchen sind nun Zyklen in den Geschäftsprozessen möglich. Somit können Pfade in der Theorie endlos lang werden. Um dies zu unterbinden können Zyklen für dieses Versuch maximal 3-mal durchlaufen werden.

In Abbildung 16 wird die prozentuale Verteilung der Pfadlängen dargestellt. Direkt erkennbar ist, dass Zyklen in den generierten Prozessen ab einer Sprungpfadwahrscheinlichkeit von mehr als 0 Prozent auftreten, da auch Pfadlängen mit mehr als 100 Aktivitäten enthalten sind. Zudem ist zu beobachten, dass mit zunehmender Sprungpfadwahrscheinlichkeit die Höhepunkte der Verteilung sinken und die Pfadlängen stärker Verteilt werden.

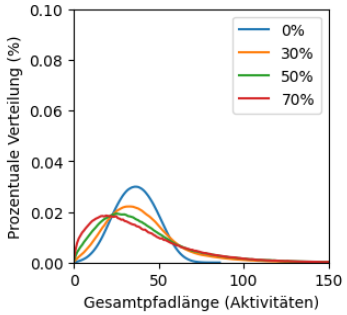


Abbildung 16: Prozentuale Verteilung der Gesamtpfadlängen

Sprungpfad %	Höhepunkt	MSE	90-Prozent Intervall
0 %	0.029932	0.124	[17, 56]
30 %	0.022137	0.057	[7, 67]
50 %	0.019317	0.038	[2, 74]
70 %	0.018537	0.034	[0, 73]

Tabelle 7: Metriken bei unterschiedlichen Sprungpfadwahrscheinlichkeit

Die Tabelle 7 enthält errechnete Metriken zu den Graphen. Enthalten sind die Wahrscheinlichkeit der am häufigsten vorkommenden Pfadlänge (*Höhepunkt*), die Abweichung zum Optimum (*MSE*) und das Intervall um den Höhepunkt in dem 90 Prozent der generierten Pfade liegen. Auch in dieser Tabelle ist zu erkennen, dass eine höhere Sprungpfadwahrscheinlichkeit zu einer geringeren Abweichung führt. Das 90 Prozent Intervall steigt stark mit zunehmender Sprungpfadwahrscheinlichkeit. Ein größeres Intervall impliziert somit stärker voneinander abweichende Pfadlängen. Da die Kurven mit zunehmender Sprungpfadwahrscheinlichkeit flacher werden, nähern sie sich dem Optimum an und die MSE sinkt.

Fazit Alle errechneten Metriken weisen daraufhin, dass Sprungpfade zu stärker variierenderen Pfadlängen führen und die Vorreingenommenheit somit sinkt.

5.4.8 Verknüpfen der Attribute

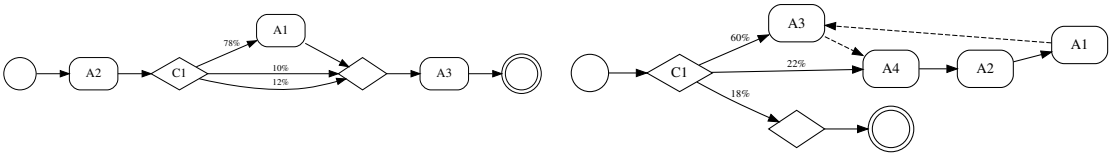
Nach der Prozesserstellung werden die zuvor generierten Attribute (siehe Unterunterabschnitt 5.4.4) mit den Aktivitäten verknüpft. Jedes Attribut enthält eine Variable, die angibt, mit wie vielen Aktivitäten es verknüpft werden soll. Entsprechend dieser Anzahl

werden zufällig Aktivitäten für jedes Attribut aus dem Prozess ausgewählt und mit diesem verknüpft. Dabei kann dieselbe Aktivität mit mehreren Attributen verknüpft werden, jedoch wird ein Attribut niemals mehrfach mit derselben Aktivität verknüpft.

5.4.9 Prozesse Validieren

Um die Algorithmen und Konzepte der Prozessgenerierung möglichst einfach zu halten, werden Zustände, die zu ungültigen Geschäftsprozessen führen, bewusst nicht berücksichtigt. Dies kann zur Entstehung von Prozessen führen, die Fehler enthalten oder in der Praxis in dieser Weise nicht auftreten können. In diesem letzten Schritt wird daher überprüft, ob die generierten Prozesse in sich valide und konsistent sind. Ein Prozess wird als valide angesehen, wenn alle der folgenden Regeln erfüllt sind:

- Eine Bedingung darf nur einen leeren Pfad enthalten. In Abbildung Abbildung 17a wird ein Prozess dargestellt, der dies nicht erfüllt..
- Es dürfen keine Endlosschleifen im Prozess enthalten sein. Dies tritt auf, wenn zwei Sprungpfade gegenseitig aufeinander verweisen (siehe Abbildung 17b).
- Pfade dürfen nicht „ins Leere laufen“ und der Endknoten muss von jedem Knoten des Prozesses erreichbar sein.



(a) (b)

Abbildung 17: Beispiele für invalide Geschäftsprozesse.

5.4.10 Visualisierung der Prozesse

Um ein Verständnis über die Struktur der zufällig generierten Prozesse zu erhalten, können die Prozesse im Prototyp visualisiert werden. Es ist möglich, bei der Generierung der Prozesse ein HTML-Protokoll zu erzeugen. In diesem sind Grafiken über alle erstellten Prozesse enthalten, sowohl valide als auch invalide. Für die fehlerhaften Prozesse wird zusätzlich der Grund angezeigt, weshalb diese als invalide gelten.

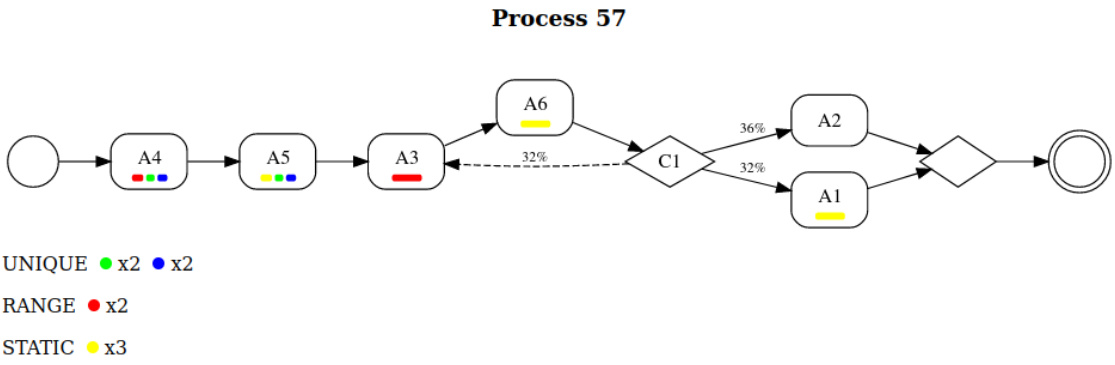


Abbildung 18: Visuelle Repräsentation eines Geschäftsprozesses aus dem Prototyp

In Abbildung 18 wird die visualisierte Grafik eines einzelnen Prozesses aus dem HTML-Protokoll dargestellt. Der runde Knoten mit einfachem Rand steht für den Startknoten und der Knoten mit doppeltem Rand für den Endknoten. Aktivitäten werden über rechteckige Knoten dargestellt, deren Namen mit A beginnen und fortlaufend nummeriert sind. Bedingungen werden durch Rauten dargestellt und enthalten ein C im Namen. Ein Knoten mit der Form einer Raute, welcher keinen Namen trägt, ist der Endknoten einer Bedingung. Sind zwei Knoten mit einem Pfeil miteinander verbunden, so folgen sie im Prozess aufeinander. Hat der Pfeil eine gestrichelte Linie, handelt es sich um die Verbindung eines Sprungknotens, der auf einen beliebigen Knoten im Prozess verweist.

Attribute sind durch farbliche Punkte in den Aktivitäten markiert. In der Abbildung existieren 2 eindeutige Attribute (grün und blau), ein sequenzielles Attribut (rot) und ein statisches Attribut (gelb). Die Zahl, welche auf die Farbe des Attributes in der Legende folgt, steht für die Anzahl der Knoten, mit der das Attribut verbunden ist. Das blaue, eindeutige Attribut ist beispielsweise mit den Aktivitäten A4 und A5 verknüpft.

5.5 Bewertung Anforderungen an den Prozessgenerator

In Unterabschnitt 5.1 werden Anforderungen an den entwickelten Prozessgenerator definiert. Diese Anforderungen stellen sicher, dass der Prozessgenerator Daten erzeugt, die für den Trainingsbestand geeignet sind.

5.5.1 Bewertung der Konfigurierbarkeit

Die Konfigurierbarkeit des Prozessgenerators ist relevant, damit gezielt Prozesse unterschiedlicher Größe und Komplexität erzeugt werden können. In Unterunterabschnitt 5.4.2, 5.4.3 und 5.4.4 werden die Konfigurationen des Generators vorgestellt. Diese ermöglichen eine Konfigurierbarkeit der Anzahl der Aktivitäten, Bedingungen und Attribute, sowie das setzen von knotenspezifischen Eigenschaften. Für viele Konfigurationsparameter können zudem Intervalle angegeben werden, aus denen bei der Prozessgeneration zufällige Werte bestimmt werden, wie die Anzahl der Ausgangspfade der Bedingungen oder die Anzahl der Verknüpfungen von Attributen. Dies erlaubt es aus derselben Konfiguration eine Vielzahl unterschiedlicher Prozesse zu erzeugen. Daher ist geforderte Konfigurierbarkeit für die Erzeugung des Trainingsbestandes erfüllt.

5.5.2 Bewertung der Unvoreingenommenheit

Eine der zentralen Anforderungen an den Prozessgenerator ist die Erstellung unvoreingenommener Prozesse. Dies soll das Risiko verringern, dass bestimmte Prozessstrukturen bevorzugt erstellt werden und somit ein unausgewogener Trainingsdatensatz für das intelligente System entsteht.

Die Unvoreingenommenheit von Geschäftsprozessen in Metriken auszudrücken, ist keine triviale Aufgabenstellung. Geschäftsprozesse sind komplexe Konstrukte. Selbst wenn zwei Prozesse die gleiche Anzahl an Bedingungen und Aktivitäten enthalten, können sie sich grundlegend in Aufbau und Struktur unterscheiden. Die Identifikation von Fall- und Aktivitäts-Id aus Abschnitt 3 basiert auf der Erkennung von Mustern im Eventlog. Die Länge und Diversität dieser Muster hängen von der Länge der Pfade in den Prozessen ab. Daher dient die Länge der Subpfade sowie die Länge der Gesamtpfade der Prozesse als primäres Merkmale zur Bewertung der Unvoreingenommenheit.

In Unterunterabschnitt 5.4.3 wird bei der Erstellung der Bedingungen die Wahrscheinlichkeit der Ausgangspfade mithilfe der Dirichlet-Verteilung berechnet. In diesem Versuch wird gezeigt, dass die erzeugten Wahrscheinlichkeiten vollkommen unvoreingenommen sind.

Bei der Erstellung der Subpfade aus Unterunterabschnitt 5.4.5 wird die Verteilung der Subpfadlängen untersucht und bestimmt, wie sich diese auf die Länge der Gesamtpfade

auswirkt. Es wird ein Algorithmus vorgestellt, der im Vergleich zu einer naiven Referenzimplementierung unvoreingenommenere Ergebnisse erzielt. Allerdings ist dennoch eine leichte Voreingenommenheit gegenüber bestimmten Pfadlängen zu erkennen

In Unterunterabschnitt 5.4.6 wird die Gesamtpfadlänge umfangreich analysiert. Die Abweichung zum Optimum wird ermittelt und als gering eingestuft, da selbst die wahrscheinlichsten Pfadlängen nur wenige Prozent über dem theoretischen Optimum liegen. Es wird auch festgestellt, dass die erzeugten Prozesse mit hoher Wahrscheinlichkeit Pfadlängen aufweisen, die stark von den bevorzugten Pfadlängen abweichen, was zu einem breiten Spektrum an potenziellen Pfadlängen in den erzeugten Prozessen führt. Auch zeigt der Versuch, dass nahezu alle möglichen Pfadlängen in den erstellten Prozessen vorkommen.

Zuletzt wird in Unterunterabschnitt 5.4.7 der Einfluss der Sprungpfade auf die Verteilung der Gesamtpfadlängen der Prozesse untersucht. Es wird festgestellt, dass die Voreingenommenheit der Prozesse mit zunehmender Sprungpfadwahrscheinlichkeit weiter abnimmt.

Aus den meisten Versuchen geht hervor, dass eine Voreingenommenheit in den erstellten Prozessen vorhanden ist, diese jedoch als *gering* eingestuft wird. Es wird zudem gezeigt, dass Konfigurationsparameter wie die Anzahl der Bedingungen oder die Sprungpfadwahrscheinlichkeit Einfluss auf die Voreingenommenheit haben. Bei größeren Prozessen mit mehr Bedingungen zeigt sich eine größere Voreingenommenheit. Um dem entgegenzuwirken, werden bei der Generation des Trainingsdatensatzes größere Wertebereiche für die Konfigurationsparameter verwendet, insbesondere bei der Erstellung großer Prozesse. Aus diesen Gründen wird die Anforderung der Unvoreingenommenheit für den Anwendungsfall dieser Arbeit als ausreichend und damit erfüllt betrachtet.

5.5.3 Bewertung der Konsistenz

Es wird die Anforderung gestellt, dass die erzeugten Prozesse in sich konsistent und korrekt sein müssen. Dies wird über zwei Mechanismen sichergestellt. In Unterunterabschnitt 5.4.9 wird ein System vorgestellt, das die generierten Prozesse nach der Erzeugung auf ihre Korrektheit prüft. Zusätzlich wird in Unterunterabschnitt 5.4.10 eine Möglichkeit zur Visualisierung der Prozesse vorgestellt. Dies ermöglicht eine manuelle Überprüfung

der erstellten Prozesse. Somit können Fehler in der Implementierung erkannt und behoben oder neue Regeln in die Prozessvalidierung aufgenommen werden. Die Anforderung an die Konsistenz wird somit erfüllt.

5.5.4 Bewertung des Determinismus

Diese Anforderung setzt voraus, dass der Prozessgenerator und die Prozess-Engine deterministisch sein müssen. Dies ermöglicht sowohl die Reproduzierbarkeit von Fehlern als auch die Replikation wissenschaftlicher Ergebnisse. Bei der Implementierung wird daher darauf geachtet, dass für alle Zufallszahlen derselbe *seed* verwendet wird und dass Programmstrukturen wie *Hashmaps* vermieden werden, da diese zu nicht deterministische Reihenfolgen führen können.

Die Prozess-Engine und der Prozessgenerator werden mit einem Versuch auf ihre Deterministik getestet. Der Prozessgenerator wird nacheinander 10-mal in einem eigenen Systemprozess aufgerufen. In jedem Systemprozess werden 100 000 Geschäftsprozesse mit derselben Konfiguration generiert, simuliert und als Eventlog abgespeichert. Durch einen bitweisen Vergleich werden die erzeugten Eventlogs miteinander verglichen. Das Ergebnis dieses Versuchs zeigt, dass alle Eventlogs den identischen Inhalt haben. Aus diesem kann gefolgert werden, dass die Generierung und Simulation der Prozesse deterministisch ist. Somit wird die Anforderung erfüllt.

5.6 Der Trainingsbestand

Die Trainingsdaten werden mithilfe der zufällig generierten Geschäftsprozesse aus Unterabschnitt 5.4 erstellt und mittels der Prozess-Engine aus Unterabschnitt 5.3 simuliert. Die auf diese Weise erzeugten Eventlogs dienen als Eingangsdaten für die Eventloganalyse aus Abschnitt 4, um daraus Kennzahlen zu errechnen. Diese Kennzahlen dienen als Trainingsdaten für das intelligente System.

Damit der Trainingsbestand möglichst divers und umfangreich ist, werden Prozesse aus vier unterschiedlichen Konfigurationen generiert. Diese unterscheiden sich sowohl in der Größe, die über die Anzahl der Knoten festgelegt ist, als auch in der Komplexität, die über die Anzahl der Bedingungen, die Wahrscheinlichkeit der Sprungpfade und die Anzahl der Attribute bestimmt wird. Die Szenarien *klein/einfach*, *groß/einfach*, *klein/komplex* und *groß/komplex* werden durch die Konfigurationen aus Tabelle 8 abgebildet.

Szenario	**klein/einfach**	**groß/einfach**	**klein/komplex**	**groß/komplex**
Aktivitäten	10-30	50-100	10-30	50-100
Bedingungen	1-5	1-5	5-10	5-20
Zweige je Bedingung	2-3	2-3	2-3	2-6
Sprungpfade	0.1	0.1	0.3	0.3
Anzahl Attribute	1-2	1-2	1-2	1-2
Verknüpfungen	1-10	1-10	1-10	1-10

Tabelle 8: Konfiguration der Szenarien des Trainingsdatensatzes

Für jede dieser Konfigurationen werden 100 000 Geschäftsprozesse erzeugt und anschließend simuliert. Jede Simulation umfasst zwischen 1 000 und 5 000 Prozessinstanzen mit einer Wahrscheinlichkeit des Rauschens von 0.001 bis 0.1 Prozent. Alle in der Prozess-Engine enthaltenen Arten des Rauschens treten mit dieser Wahrscheinlichkeit auf.

Aus jedem Eventlog wird das korrekte und ein zufälliges falsches Kandidatenpaar ausgewählt. Diese werden analysiert und ihre Kennzahlen berechnet. Die Kennzahlen werden zusammen mit der Klassifizierung, die angibt, ob das Kandidatenpaar korrekt oder inkorrekt ist, als JSON abgespeichert. Das Ergebnis ist ein Datensatz, der aus 400 000 Datenpunkten besteht, von denen 200 000 der richtigen Fall- und Aktivitäts-Id entsprechen.

6 Das intelligente System

Das intelligente System wird für die Auswertung der in Abschnitt 4 errechneten Kennzahlen verwendet, um das Kandidatenpaar zu identifizieren, dass die korrekte Fall- und Aktivitäts-Id enthält. Dieses Problem kann als Klassifizierungsproblem betrachtet werden, bei dem aus allen Kandidatenpaaren dasjenige ausgewählt wird, das die korrekte Fall- und Aktivitäts-Id enthält.

6.1 Technologie des intelligenten Systems

Im Kontext des Maschinellen Lernens existieren eine Vielzahl an Technologien, die auf Klassifizierungsprobleme ausgelegt sind [14]. Unter diesen befinden sich neuronale Netzwerke, Random Forests, Petri-Netze und lineare Regression. In einer Studie werden 179 Klassifizierungstechnologien unter Verwendung von 121 Datensätzen miteinander verglichen [14]. In über 90 Prozent der Datensätze zeigen Random Forests die besten Resultate

[14]. Dicht gefolgt werden diese von neuronalen Netzwerken. Neurale Netzwerke weisen in dieser Studie im Durchschnitt zwar schlechtere Ergebnisse auf, erzielen jedoch bei komplexen Aufgabenstellungen und Sachverhalten bessere Resultate als Random Forests [14].

Beide Technologien werden in dieser Arbeit als Kandidaten für das intelligente System betrachtet. Final wird sich für Random Forests entschieden, da sie im Vergleich zu neuronalen Netzwerken folgende Vorteile bieten:

- **Kürzere Lernzeit:** Die Trainingsphase von Random Forests ist typischerweise kürzer als die von neuronalen Netzwerken, da diese weniger rechenintensiv ist und stärker parallelisiert werden kann [2].
- **Einfachere Modelloptimierung:** Random Forests weisen im Vergleich zu neuronalen Netzwerken weniger Hyperparameter auf. Somit wird der Prozess der Modellanpassung und -optimierung beschleunigt und vereinfacht [2].
- **Datenvorverarbeitung:** Im Vergleich zu Random Forests benötigen neuronale Netzwerke die Eingangsdaten in einer vorgegebenen Form, um optimale Ergebnisse zu erzielen. Diese müssen in numerischer Form vorliegen und werden typischerweise auf den Wertebereich $[-1, 1]$ normalisiert [2]. Dies ist für die Kennzahlen der Kandidatenpaare nur bedingt möglich, da diese keine festen und erwarteten Wertebereiche enthalten . Ein Beispiel ist die Anzahl der Fall-Ids die im Eventlog auftreten. In einem Eventlog können theoretisch beliebig viele Prozessinstanzen enthalten sein, somit existiert kein oberes Limit.

Diese Gründe sorgen für einen schnelleren Entwicklungsprozess, da weniger Zeit für das Anlernen der Modelle und deren Optimierung benötigt wird. Dies ermöglicht es, im Rahmen dieser Arbeit mehrere Klassifizierungsstrategien zu erarbeiten und zu validieren. Diese unterscheiden sich in der Art und Weise, wie das korrekte Kandidatenpaar identifiziert wird.

Implementiert wird der Random Forest in Python mithilfe der *scikit-learn* Bibliothek [17]. Diese bietet einfache Schnittstellen für das Anlernen und das Evaluieren der Modelle.

6.2 Hyperparameter und Modelloptimierung

Eine der großen Herausforderungen im maschinellen Lernen ist die Wahl geeigneter Hyperparameter, da diese maßgeblich für den Ergfolg der Modelle verantwortlich sind [23]. Die Bibliothek *scikit-learn* bietet mehrere Verfahren zur automatisierten Optimierung dieser Parameter. Verwendet wird die *Bayes'sche Optimierung*, da diese in der *Black-Box Optimization Challenge* aus 2020 die besten Ergebnisse erzielt [23]. Alle der besten 20 Teilnehmerteams dieses Wettbewerbs verwenden in ihrer Implementierung die *Bayes'sche Optimierung* oder Derivate von dieser Optimierung.

Für die *Bayessche Optimierung* können Suchräume für jeden Hyperparameter des Random Forests bestimmt werden. Bei der Optimierung werden automatisiert Werte aus diesen Suchräumen ermittelt, die zu dem Modell mit der höchsten Genauigkeit führt.

```
{
    "n_estimators": [10, 500],
    "max_depth": [1, 100],
    "min_samples_split": [2, 10],
    "min_samples_leaf": [1, 10]
}
```

Auflistung 5: Konfiguration für die Bayes'sche Optimierung der Random Forests

In Auflistung 5 wird die verwendete Konfiguration für den Optimierungsalgorithmus dargestellt. Diese enthält folgende Parameter:

- **n_estimators:** Dieser Parameter bestimmt die Anzahl der Entscheidungsbäume im Random Forest. Die Wahl eines Bereichs von 10 bis 100 ermöglicht es, sowohl Modelle mit einer geringeren als auch mit einer höheren Anzahl von Bäumen zu testen. Nach [19] haben insbesondere die ersten 100 Entscheidungsbäume den größten Einfluss auf die Ergebnisse.
- **max_depth:** Dieser Parameter bestimmt die Tiefe der Entscheidungbäume. Eine Spanne von 1 bis 20 deckt sowohl sehr flache als auch sehr tiefe Bäume ab. Flache Bäume neigen weniger zu Overfitting. Im Gegensatz dazu können tiefere Bäume komplexere Muster in den Daten erfassen und erlernen [19].
- **min_samples_split und min_samples_leaf:** Diese Parameter legen die Mindestanzahl der Datenpunkte fest, die erforderlich sind, um einen Knoten zu spalten

(*min_ samples_ split*), bzw. die Mindestanzahl an Datenpunkten, die ein Blatt enthalten muss (*min_ samples_ leaf*). Kleinere Werte können zu detaillierteren und komplexeren Modellen führen, da sie feinere Strukturen in den Daten erkennen. Größere Werte hingegen reduzieren die Modellkomplexität und tragen dazu bei, Overfitting zu verhindern. Die gewählten Werte basieren auf den Empfehlungen der Dokumentation der scikit-learn Bibliothek [17].

Die Konfiguration aus Auflistung 5 wird für alle Klassifizierungsstrategien verwendet, um eine Vergleichbarkeit dieser herzustellen. Alle Klassifizierungsstrategien können somit auf Modelle der gleichen Komplexität zugreifen. Keine Strategie hat somit einen systematischen Vorteil, der zu besseren Ergebnissen führen könnte. Ein Nachteil dieses Ansatzes besteht darin, dass die Klassifizierungsstrategien möglicherweise nicht die theoretisch optimalen Ergebnisse liefern, da sie durch diese einheitliche Konfiguration limitiert sind. Dieser Nachteil wird jedoch im Hinblick auf die gewährleistete Vergleichbarkeit der Ergebnisse in Kauf genommen.

6.3 Die Testdatensätze

Zur Validierung des Prototyps werden mehrere Datensätze verwendet. Diese umfassen unterschiedliche Szenarien, um die Robustheit und Zuverlässigkeit der Modelle zu überprüfen. Es wird zwischen den Train-Test-Split und den PLG-Datensätzen unterschieden.

6.3.1 Train-Test-Split

Aus dem in Unterabschnitt 5.6 vorgestellten Trainingsbestand werden 10 Prozent nicht für das Training der Modelle verwendet, sondern dienen einer ersten Evaluation. Diese Daten werden bereits im Trainingsprozess des Modells eingesetzt, um den Fortschritt des Trainings zu überwachen. Der Datensatz umfasst Kennzahlen von 40 000 Kandidatenpaaren, von denen 20 000 den korrekten Fall- und Aktivitäts-Ids entsprechen.

Dieser Datensatz allein reicht nicht aus, um verlässliche Aussagen über die Genauigkeit der Modelle zu treffen. Diese Daten stammen, ebenso wie die Trainingsdaten, aus dem Prozessgenerator, daher besteht die Möglichkeit, dass das Modell charakteristische Merkmale des Generators erlernt, und nicht die tatsächliche Identifikation der Fall- und

Aktivitäts-Id. Aus diesem Grund werden für die finale Evaluation der Modelle zusätzlich Datensätze aus Fremdsystemen verwendet.

6.3.2 PLG-Datensätze

In Abschnitt 5 wird bereits auf die unzureichende Anzahl frei zugänglicher Trainingsdaten eingegangen. Dieselbe Problematik besteht ebenfalls bei der Suche nach geeigneten Testdatensätzen. In Unterabschnitt 5.2 wird der Prozessgenerator PLG vorgestellt und die Gründe erläutert, weshalb dieser für die Erzeugung der Trainingsdaten nicht geeignet ist. Zusammengefasst beruht die Entscheidung auf folgenden Gründen:

- Die Größe und Komplexität der Prozesse kann nur bedingt konfiguriert werden [12, 11].
- Die Unvoreingenommenheit der erzeugten Prozesse ist in keiner Veröffentlichung belegt [12, 11].
- Die Attribute weisen keine komplexen Wertwiederholungen auf [12, 11].

Diese Gründe werden jedoch als weniger relevant für den Einsatz als Prozessgenerator für die Testdaten betrachtet. Für die Evaluation werden deutlich weniger Datenpunkte als für das Training benötigt. Daher können die erstellten Prozesse manuell geprüft und kategorisiert werden. Die Unvoreingenommenheit der Prozesse spielt ebenfalls eine geringere Rolle, da nicht die Gefahr besteht, dass das Modell diese erlernt. Zudem kann bei der manuellen Selektion der Prozesse auf eine Unvoreingenommenheit geachtet werden.

Die unzureichende Komplexität der Attribute stellt jedoch einen erheblichen Nachteil für die Evaluation der Modelle dar. In PLG können Attribute nur mit einer Aktivität verknüpft werden, wodurch komplexe Wertwiederholungen nicht möglich sind. Daher wird eine Erweiterung für PLG implementiert, die diese Einschränkung aufhebt.

PLG Attribute Bei der Erzeugung eines Prozesses über die Benutzeroberfläche von PLG kann ein Wahrscheinlichkeitswert angegeben werden, mit dem eine Aktivität mit einem verknüpften Attribut erzeugt wird. Dieses Attribut erhält einen Namen und einen festen Wert. Während der Simulation wird der Name des Attributs als Spalte in den Eventlog aufgenommen. Der Wert dieser Spalte bleibt leer, bis die Aktivität, die mit dem Attribut verknüpft ist, ausgeführt wird. Bei der Ausführung der Aktivität wird der

Wert des Attributs in die Spalte mit dem entsprechenden Attributnamen eingetragen. Dieses Verhalten entspricht dem statischen Attribut in der entwickelten Prozess-Engine (siehe Unterunterabschnitt 5.3.6), mit der Einschränkung, dass das ein Attribut nur mit einer Aktivität verknüpft werden kann.

PLG Erweiterung Die entwickelte Erweiterung wird nach der Prozesserstellung aktiv. Zunächst werden alle Attribute nacheinander durchlaufen und x Kopien jedes Attributs erstellt. Die Kopien erhalten denselben Namen wie das Original, jedoch einen eigenen zufälligen Wert. Die Anzahl x ist ein zufälliger Wert, der für jedes Attribut aus einem zuvor definierten Intervall ausgewählt wird. Dieses Intervall kann bei der Prozesserstellung in der Benutzeroberfläche angegeben werden.

Die kopierten Attribute werden anschließend an zufällige Knoten im Prozess angehängt. Die Anzahl der Knoten wird ebenfalls über ein Intervall bestimmt, dass in der Benutzeroberfläche gesetzt werden kann. Da die kopierten Attribute denselben Namen wie das Original besitzen, schreiben diese bei der Simulation alle in dieselbe Spalte des Eventlogs. Da die kopierten Attribute unterschiedliche Werte besitzten, entstehen komplexere Wertwiederholungen im Eventlog.

Es wird bewusst darauf verzichtet, dieselben Attributarten der eigenen Prozess-Engine zu implementieren. Dadurch unterscheiden sich die Attribute im Verhalten zwischen Trainings- und Testdaten. Dies ist bei der Evaluation erwünscht, da getestet werden soll, ob die Modelle auch auf Daten aus Fremdsystemen anwendbar sind.

Die Datensätze Aus PLG werden mehrere Datensätze erstellt, welche die Modelle auf unterschiedliche Kriterien untersuchen. Diese können in 4 Kategorien eingeteilt werden.

- **Basis-Datensätze:** Diese Kategorie umfasst drei Datensätze, die jeweils 100 Prozesse für die Szenarien *klein/einfach*, *klein/komplex* und *groß/komplex* enthalten. Diese Szenarien entsprechen den Szenarien der Trainingsdaten, die in Unterabschnitt 5.6 beschrieben werden. Das Szenario *groß/einfach* kann nicht durch PLG abgebildet werden, da es nicht möglich ist, Prozesse mit 50-100 Aktivitäten zu erzeugen, die weniger als 5-10 Bedingungen enthalten. Grund hierfür ist der begrenzte Einfluss auf die Prozesskonfiguration, da Prozesse mit mehr als 50 Aktivitäten durch PLG immer mehr als 10 Bedingungen aufweisen. Es wird angenommen, dass diese Datensätze die geringste Herausforderung für die Modelle darstellt, da diese

den Trainingsdaten am ähnlichsten sind. Sie dienen der grundlegenden Überprüfung der Modellfunktionalität und ob diese die Fall- und Aktivitäts-Id in einfachen Szenarien identifizieren können.

- **Rausch-Datensätze:** Diese Kategorie umfasst sechs Datensätze, von denen jeder 100 Prozesse enthält. Im Vergleich zu den Basis-Datensätzen enthalten diese jedoch Rauschen. Für die Szenarien *klein/einfach*, *klein/komplex* und *groß/komplex* werden jeweils Eventlogs mit 10 Prozent und 30 Prozent Rauschen erstellt. Die Datensätze dieser Kategorie untersuchen, wie gut die Modelle mit moderatem und extremem Rauschen umgehen können.

- **Parallel-Datensätze:** Im Gegensatz zur eigenen Prozess-Engine ermöglicht PLG die Erzeugung von Parallelen-Gateways und damit die Modellierung von Nebenläufigkeit. In dieser Kategorie werden die Hälfte der Bedingungen durch Parallel-Gateways ersetzt. Für die Szenarien *klein/einfach*, *klein/komplex* und *groß/komplex* werden jeweils 100 Prozesse erstellt. Mit den Datensätzen dieser Kategorie soll untersucht werden, welchen Einfluss Nebenläufigkeit auf die Leistung der Modelle hat.

- **Exoten-Datensätze:** Diese Kategorie besteht aus zwei Datensätzen, die jeweils 100 Prozesse enthalten. Im ersten Datensatz werden besonders große Prozesse abgebildet, die zwischen 200 und 500 Aktivitäten und 40 bis 200 Bedingungen umfassen. Diese Prozesse sind mindestens doppelt so groß und komplex wie die größten Prozesse der Trainingsdaten. Im zweiten Datensatz werden Prozesse erstellt, die dem *groß/komplex*-Szenario entsprechen, jedoch werden diese mit 25 000 Prozessinstanzen simuliert. Dadurch enthalten diese Eventlogs mindestens das Fünffache der Prozessinstanzen im Vergleich zu den Trainingsdaten. Es wird erwartet, dass beide Datensätze die größte Herausforderung für die Modelle darstellt, da sie Szenarien Testen, die erheblich von den Trainingsdaten abweichen.

In Tabelle 9 werden all PLG-Testdatensätze mit Namen, Kategorie und zugehörigem Szenario aufgeführt. Die Namen der Datensätze werden in der folgenden Evaluation und insbesondere in den Schaubildern und Tabellen verwendet. Jede Zeile der Tabelle repräsentiert einen Datensatz mit jeweils 100 Prozessen. Der gesamte PLG-Testdatensatz umfasst somit 1 400 Prozesse. Alle Prozesse, mit Ausnahme des Datensatzes *manyInstances*, werden mit 2 500 Prozessinstanzen simuliert.

Datensatz	Kategorie	Szenario
smallSimple	Basis-Datensatz	klein/einfach
smallComplex	Basis-Datensatz	klein/komplex
largeComplex	Basis-Datensatz	groß/komplex
smallSimpleNoise10	Rausch-Datensatz (10 %)	klein/einfach
smallComplexNoise10	Rausch-Datensatz (10 %)	klein/komplex
largeComplexNoise10	Rausch-Datensatz (10 %)	groß/komplex
smallSimpleNoise30	Rausch-Datensatz (30 %)	klein/einfach
smallComplexNoise30	Rausch-Datensatz (30 %)	klein/komplex
largeComplexNoise30	Rausch-Datensatz (30 %)	groß/komplex
smallSimpleParallel	Parallel-Datensatz	klein/einfach
smallComplexParallel	Parallel-Datensatz	klein/komplex
largeComplexParallel	Parallel-Datensatz	groß/komplex
extraLarge	Exot-Datensatz	extra groß
manyInstances	Exot-Datensatz	groß/komplex

Tabelle 9: Namen, Kategorien und Szenarien der PLG-Datensätze

6.4 Klassifizierungsstrategien

In dieser Arbeit werden verschiedene Klassifizierungsstrategien untersucht, um das korrekte Kandidatenpaar zu bestimmen. Diese Strategien unterscheiden sich hinsichtlich der Informationen, die den Modellen zugeführt werden, sowie in der Art und Weise, wie die Prognosen interpretiert werden.

6.4.1 Direktprognose

Bei der Direktprognose wird direkt vorhergesagt, ob ein Kandidatenpaar die korrekte Fall- und Aktivitäts-Id enthält. Dazu werden die Kennzahlen eines Kandidatenpaars dem Modell vorgelegt, das diese auswertet und einen Wahrscheinlichkeitswert prognostiziert. Dieser gibt an, wie wahrscheinlich es ist, dass das Paar dem korrekten Kandidatenpaar entspricht. Dieser Vorgang wird für alle Kandidatenpaare wiederholt. Das Paar mit dem höchsten Wahrscheinlichkeitswert wird als das finale Ergebnis betrachtet.

Eingangsvektor Alle normalen Kennzahlen aus Unterunterabschnitt 4.3.4 sowie deren bereinigte Varianten ohne Rauschen werden für den Eingangsvektor verwendet. Dieser umfasst somit 12 Merkmale.

Ausgangsvektor Der Ausgangsvektor besteht aus einem booleschen Wert, der angibt, ob es sich bei dem Kandidatenpaar um das korrekte Paar handelt. Falls das Kandidatenpaar korrekt ist, enthält der Ausgangsvektor den Wert *true*, andernfalls den Wert *false*.

6.4.2 Zweiklassenvergleich

Der Zweiklassenvergleich orientiert sich am Ligasystem im Sport. Alle Kandidatenpaare werden in eine Liga eingeteilt. Innerhalb der Liga werden alle Kandidatenpaare miteinander verglichen.

Das Modell bestimmt bei dem Vergleich den *Sieger*, indem es prognostiziert, welches der beiden Paare stärkere charakteristische Merkmale der Fall- und Aktivitäts-Ids aufweist. Der Sieger des Vergleichs erhält einen Punkt. Um eine Voreingenommenheit gegenüber der Position im Vergleich zu vermeiden, treten alle Kandidatenpaare zweimal gegeneinander an, jeweils mit vertauschter Vergleichsreihenfolge. Das Kandidatenpaar, das nach allen Vergleichen innerhalb der Liga die meisten Punkte erzielt, wird als Sieger der Liga betrachtet.

Nachteil dieser Vergleichsprognose ist die Laufzeitkomplexität. Da alle Kandidatenpaare zweimal miteinander verglichen werden, beträgt die Komplexität $O(2 \cdot |K|^2 - |K|)$, wobei $|K|$ der Anzahl der Kandidatenpaare entspricht. Gemäß Unterabschnitt 4.2 steigt die Anzahl der Kandidatenpaare quadratisch mit der Anzahl der Attribute im Eventlog. Daher müssen mindestens $|A|^4$ Vergleiche durchgeführt werden, wobei $|A|$ die Anzahl der Attribute ist. Dies kann bei großen Eventlogs mit vielen Attributen zu sehr langen Laufzeiten führen.

Eine Möglichkeit, die Komplexität zu reduzieren, besteht darin, die Kandidatenpaare in mehrere Ligen aufzuteilen. In jeder Liga wird der Sieger ermittelt, der in die nächsthöhere Liga aufsteigt. Aus diesen höheren Liegen werden ebenfalls neue Sieger bestimmt, die weiter aufsteigen. Dieser Prozess wird fortgesetzt, bis der finale Sieger feststeht. Die geringste Laufzeitkomplexität entsteht, wenn nur Ligen mit je 2 Kandidatenpaaren gebildet werden. Diese wird auf $O(2 \cdot |K| \cdot log(|K|) - |K|)$ reduziert.

Ein wesentlicher Nachteil dieser Optimierung besteht darin, dass das korrekte Kandidatenpaar bei einer Fehlentscheidung des Modells möglicherweise frühzeitig ausgeschlossen wird. Angenommen, es gibt ein einzelnes falsches Kandidatenpaar, das vom Modell als

wahrscheinlicher angesehen wird als das korrekte Paar. Bei der Direktprognose könnte sich das korrekte Kandidatenpaar dennoch in der finalen Bewertung gut positionieren, da es direkt auf Grundlage seiner Wahrscheinlichkeit bewertet wird. Dies ist beim Zweiklassenvergleich nicht gegeben. Die finale Position des korrekten Kandidatenpaars hängt stark davon ab, in welcher Liga es gegen das falsche Kandidatenpaar antritt. Eine Fehlentscheidung in einer unteren Liga kann dazu führen, dass das korrekte Paar frühzeitig ausgeschlossen wird, obwohl es möglicherweise im Vergleich mit allen anderen Kandidatenpaaren als Sieger hervorgegangen wäre.

Diese Optimierung für die Evaluation nicht verwendet. Da die Eventlogs der Testdaten in der Regel weniger als 10 Attribute aufweisen, kann die Laufzeitkomplexität vernachlässigt werden. Zudem führt die Verwendung einer großen Liga mit allen Kandidatenpaaren zu mehr Vergleichen und damit zu einer höheren Anzahl von Datenpunkten, die für die Auswertungsstatistiken herangezogen werden können.

Für den Lernprozess werden aus den Trainingsdaten Paare aus jeweils einem korrekten und einem falschen Kandidatenpaar gebildet. Dies reduziert die Größe des Trainingsdatensatzes um die Hälfte. Insgesamt können daher 200 000 Paare für den Trainingsprozess gebildet werden.

Eingangsvektor Der Eingangsvektor kann in zwei Bereiche unterteilt werden. Wenn die Kandidatenpaare k_1 und k_2 miteinander verglichen werden, enthalten die ersten 12 Felder des Eingangsvektors alle Kennzahlen von k_1, während die folgenden 12 Felder die Kennzahlen von k_2 abbilden.

Ausgangsvektor Der Ausgangsvektor besteht aus einem booleschen Wert, der angibt, welches der beiden Kandidatenpaare am Eingang korrekt ist. Bei einem Vergleich der Kandidatenpaare k_1 und k_2 bedeutet der Wert *true* im Ausgangsvektor, dass k_1 das korrekte Kandidatenpaar ist, während *false* vorhersagt, dass k_2 das korrekte Kandidatenpaar ist.

6.4.3 Dreiklassenvergleich

Der Dreiklassenvergleich folgt einem ähnlichen Prinzip wie der Zweiklassenvergleich. Auch hier werden Ligen gebildet, in denen die Kandidatenpaare miteinander verglichen werden.

Die Besonderheit des Dreiklassenvergleichs liegt darin, dass ein zusätzliches Ergebnis berücksichtigt wird. Während der Zweiklassenvergleich lediglich bestimmt, welches der beiden Kandidatenpaare stärkere Merkmale der Fall- und Aktivitäts-Id aufweist, ermöglicht der Dreiklassenvergleich eine dritte Option. Diese besagt, dass keines der beiden zu vergleichenden Kandidatenpaare die korrekte Fall- und Aktivitäts-Id enthält. Ist dies der Fall, so wird beiden Kandidatenpaaren ein Punkt abgezogen.

Dieser Ansatz bietet zwei wesentliche Vorteile. Erstens beeinflussen sich die falschen Kandidatenpaare gegenseitig negativ. Somit fallen einzelne Fehlentscheidungen, bei denen das korrekte Kandidatenpaar fälschlicherweise als nicht korrekt bewertet wird, weniger stark ins Gewicht. Dies kann zu einer höheren Robustheit gegenüber Fehlentscheidungen führen. Zweiten können mehr Datenpunkte für das Training verwendet werden. Neben den bestehenden Paaren aus korrekten und falschen Kandidatenpaaren können auch neue Paare aus zwei falschen Kandidatenpaaren gebildet werden. Dies erhöht die Menge der verfügbaren Trainingsdaten um 25 Prozent.

Eingangsvektor Der Eingangsvektor ist identisch mit dem des Zweiklassenvergleichs. Er besteht aus zwei Bereichen, die jeweils die Kennzahlen der zu vergleichenden Kandidatenpaare enthalten.

Ausgangsvektor Der Ausgangsvektor besteht aus einem Feld, dass das zu klassifizierende Ergebnis angibt. Beim Vergleich der Kandidatenpaare k_1 und k_2 bedeutet ...

- ein Wert von 0 im Ausgangsvektor, dass k_1 das korrekte Kandidatenpaar ist.
- ein Wert von 1 im Ausgangsvektor, dass k_2 das korrekte Kandidatenpaar ist.
- ein Wert von 2 im Ausgangsvektor, dass weder k_1 noch k_2 das korrekte Kandidatenpaar ist.

6.5 Evaluation

In diesem Unterabschnitt werden die drei vorgestellten Klassifizierungsstrategien anhand der Testdatensätze evaluiert. Zunächst werden die Ergebnisse jeder Klassifizierungsstrategie vorgestellt und auf Besonderheiten in den Resultaten eingegangen. Anschließend werden die Ergebnisse der Strategien miteinander verglichen.

Datensatz	Genauigkeit	Richtig Klassifiziert	Top 3	Richtige Fall-Id	Richtige Aktivitäts-Id
Basis-Datensätze	**80.55 %**	**92.00 %**	**99.33 %**	**93.00 %**	**92.00 %**
smallSimple	59.25 %	79.00 %	98.00 %	79.00 %	79.00 %
smallComplex	85.78 %	98.00 %	100.00 %	100.00 %	98.00 %
largeComplex	96.61 %	99.00 %	100.00 %	100.00 %	99.00 %
Rausch-Datensätze-10	**36.57 %**	**67.14 %**	**98.32 %**	**100.00 %**	**67.14 %**
...	...	...	..	...	..
Durchschnitt	**53.76 %**	**69.49 %**	**92.14 %**	**92.71 %**	**69.49 %**

Tabelle 10: Beispiel der Auswertung von PLG-Datensätzen

In Tabelle 10 wird eine beispielhafte Auswertung einer Strategie dargestellt. Eine fett markierte Zeile repräsentiert die Durchschnittswerte einer Datensatzkategorie (siehe Unterunterabschnitt 6.3.2). Diese wird gefolgt von den individuellen Datensätzen dieser Kategorie. Die letzte Zeile der Tabelle zeigt den Gesamtdurchschnitt aller Kategorien an. Alle Kategorien werden dabei gleich gewichtet, auch wenn weniger Datensätze enthalten sind.

Die Spalten der Tabelle haben folgende Bedeutungen:

- **Datensatz:** Enthält den Namen des getesteten Datensatzes bzw. der Kategorie. Eine vollständige Liste aller Datensätze ist in Tabelle 9 zu finden.
- **Genauigkeit:** Gibt die Genauigkeit an, mit der das korrekte Kandidatenpaar identifiziert wird. Dieser Wert entspricht dem Durchschnitt, der über alle Prozesse eines Datensatzes gebildet wird. Die Genauigkeit mit der die falschen Kandidatenpaare klassifiziert werden, fließt nicht in diese Wertung ein. Dies ermöglicht es, dass die Genauigkeit für alle Klassifizierungsstrategien auf dieselbe Weise berechnet wird.
- **Richtig Klassifiziert:** Enthält den Prozentsatz der Prozesse, mit dem die erste Prognose der Modelle dem korrekten Kandidatenpaar entspricht.
- **Top 3:** Gibt den Prozentsatz an, mit dem sich das korrekte Kandidatenpaar unter den besten 3 Prognosen befindet.
- **Richtige Fall-Id:** Listet die prozentuale Häufigkeit auf, mit de das erste prognostizierte Kandidatenpaar die korrekte Fall-Id enthält.
- **Aktivitäts-Id:** Gibt den Prozentsatz an, mit dem das erste prognostizierte Kandidatenpaar die korrekte Aktivitäts-Id enthält.

6.5.1 Direktprognose

	Vorhersage negativ	Vorhersage positiv
Tatsächlich negativ	19485 / 49.96 %	15 / 0.04 %
Tatsächlich positiv	135 / 0.35 %	19365 / 49.65 %

Tabelle 11: Wahrheitsmatrix der Direktprognose auf dem Train-Test-Split Datensatz

Train-Test-Split In Tabelle 11 wird die Wahrheitsmatrix der Direktprognose für den *Train-Test-Split*-Datensatz dargestellt. Mit einer Genauigkeit von 99.96 Prozent werden die Kandidatenpaare dieses Datensatzes korrekt klassifiziert. Auffällig ist, dass das Modell eher dazu neigt, das korrekte Kandidatenpaar falsch zu prognostizieren, als ein falsches korrekt zu klassifizieren.

Datensatz	Genauigkeit	Richtig Klassifiziert	Top 3	Richtige Fall-Id	Richtige Aktivitäts-Id
Basis-Datensätze	**80.55 %**	**92.00 %**	**99.33 %**	**93.00 %**	**92.00 %**
smallSimple	59.25 %	79.00 %	98.00 %	79.00 %	79.00 %
smallComplex	85.78 %	98.00 %	100.00 %	100.00 %	98.00 %
largeComplex	96.61 %	99.00 %	100.00 %	100.00 %	99.00 %
Rausch-Datensätze-10	**36.57 %**	**67.14 %**	**98.32 %**	**100.00 %**	**67.14 %**
smallSimpleNoise10	43.95 %	87.00 %	99.00 %	100.00 %	87.00 %
smallComplexNoise10	34.18 %	71.00 %	99.00 %	100.00 %	71.00 %
largeComplexNoise10	31.57 %	43.43 %	96.97 %	100.00 %	43.43 %
Rausch-Datensätze-30	**31.81 %**	**62.14 %**	**98.66 %**	**100.00 %**	**62.14 %**
smallSimpleNoise30	32.88 %	81.00 %	100.00 %	100.00 %	81.00 %
smallComplexNoise30	30.97 %	62.00 %	99.00 %	100.00 %	62.00 %
largeComplexNoise30	31.57 %	43.43 %	96.97 %	100.00 %	43.43 %
Parallel-Datensätze	**85.41 %**	**94.00 %**	**100.00 %**	**99.67 %**	**94.00 %**
smallSimpleParallel	71.73 %	92.00 %	100.00 %	99.00 %	92.00 %
smallComplexParallel	89.25 %	97.00 %	100.00 %	100.00 %	97.00 %
largeComplexParallel	95.24 %	93.00 %	100.00 %	100.00 %	93.00 %
Exoten-Datensätze	**24.86 %**	**13.50 %**	**50.50 %**	**60.00 %**	**13.50 %**
extraLarge	36.92 %	26.00 %	64.00 %	100.00 %	26.00 %
manyInstances	12.79 %	1.00 %	37.00 %	20.00 %	1.00 %
Durchschnitt	**53.76 %**	**69.49 %**	**92.14 %**	**92.71 %**	**69.49 %**

Tabelle 12: Ergebnisse der Direktprognose auf den PLG-Datensätzen

PLG-Datensätze In Tabelle 12 werden die Ergebnisse der Direktprognose auf den PLG-Datensätzen dargestellt.

Die besten Ergebnisse werden auf den Parallel-Datensätzen mit 85 Prozent Genauigkeit und 94 Prozent richtiger Klassifizierung erzielt. Am schlechtesten können die Exoten-Datensätze erkannt werden. Diese weisen sowohl die niedrigste Genauigkeit (25 Prozent)

als auch die niedrigste Anzahl an richtig klassifizierten Kandidatenpaaren (13 Prozent) auf. Besonders der *manyInstances* Datensatz wird schlecht klassifiziert, da nur 1 Prozent der korrekten Kandidatenpaare erkannt werden.

Werden die unterschiedlichen Szenarien miteinander verglichen, fällt auf, dass keines konstant bessere Ergebnisse im Vergleich zu den anderen erzielt. Beispielsweise liefert bei den Basis-Datensätzen das Szenario *groß/komplex* die beste Klassifizierung, während bei den Parallel-Datensätzen *klein/komplex* die höchsten Werte enthält.

Rauschen stellt die Direktprognose vor eine erhebliche Herausforderung. Sowohl bei 10 Prozent als auch bei 30 Prozent Rauschen sinkt die Genauigkeit um mehr als 50 Prozent. Dies führt zu vielen falsch klassifizierten Kandidatenpaaren.

Sehr gute Ergebnisse können dagegen erzielt werden, wenn nicht nur das prognostizierte, sondern die besten drei Prognosen betrachtet werden. In diesem Szenario kann durchschnittlich zu 93 Prozent das korrekte Kandidatenpaar identifiziert werden. Werden die Exoten-Datensätze aus dieser Analyse vernachlässigt, steigt dieser Wert auf 99 Prozent.

Der Direktprognose fällt es zudem generell leichter, die korrekte Fall-Id (93 Prozent) zu bestimmen, als die korrekte Aktivitäts-Id (69 Prozent).

6.5.2 Zweiklassenvergleich

	Vorhersage negativ	Vorhersage positiv
Tatsächlich negativ	19446 / 49.86 %	19 / 0.05 %
Tatsächlich positiv	17 / 0.04 %	19518 / 50.05 %

Tabelle 13: Wahrheitsmatrix des Zweiklassenvergleichs auf dem Train-Test-Split Datensatz

Train-Test-Split Für den Zweiklassenvergleichs wird auf dem Train-Test-Split eine hohe Genauigkeit erzielt, bei der 99.91 Prozent der Kandidatenpaare korrekt klassifiziert werden. Aus dem Schaubild wird ein geringes Ungleichgewicht zwischen den korrekten und falschen Kandidatenpaaren sichtbar. Dies ist darauf zurückzuführen, dass bei der Erstellung der Paare für den Vergleich zufällig entschieden wird, an welcher Position das korrekte Kandidatenpaar platziert wird. Steht das korrekte Paar an erster Stelle, wird ein *Tatsächlich positiv*-Datenpunkt gebildet, andernfalls ein *Tatsächlich Negativ*-Datenpunkt. Dieses Ungleichgewicht wird jedoch als sehr gering betrachtet, da dieses nur 0.09 Prozent beträgt.

In den Ergebnissen kann keine Voreingenommenheit gegenüber der Position des korrekten Kandidatenpaars im Vergleich festgestellt werden, da sich die inkorrekten Prognosen für *Tatsächlich positiv* und *Tatsächlich negativ* lediglich um zwei Datenpunkte unterscheiden.

Datensatz	Genauigkeit	Richtig Klassifiziert	Top 3	Richtige Fall-Id	Richtige Aktivitäts-Id
Basis-Datensätze	**91.08 %**	**99.67 %**	**100.00 %**	**100.00 %**	**99.67 %**
smallSimple	81.25 %	99.00 %	100.00 %	100.00 %	99.00 %
smallComplex	93.16 %	100.00 %	100.00 %	100.00 %	100.00 %
largeComplex	98.84 %	100.00 %	100.00 %	100.00 %	100.00 %
Rausch-Datensätze-10	**65.78 %**	**100.00 %**	**100.00 %**	**100.00 %**	**100.00 %**
smallSimpleNoise10	71.75 %	100.00 %	100.00 %	100.00 %	100.00 %
smallComplexNoise10	65.03 %	100.00 %	100.00 %	100.00 %	100.00 %
largeComplexNoise10	60.57 %	100.00 %	100.00 %	100.00 %	100.00 %
Rausch-Datensätze-30	**59.28 %**	**100.00 %**	**100.00 %**	**100.00 %**	**100.00 %**
smallSimpleNoise30	58.71 %	100.00 %	100.00 %	100.00 %	100.00 %
smallComplexNoise30	58.55 %	100.00 %	100.00 %	100.00 %	100.00 %
largeComplexNoise30	60.57 %	100.00 %	100.00 %	100.00 %	100.00 %
Parallel-Datensätze	**93.16 %**	**100.00 %**	**100.00 %**	**100.00 %**	**100.00 %**
smallSimpleParallel	86.09 %	100.00 %	100.00 %	100.00 %	100.00 %
smallComplexParallel	95.59 %	100.00 %	100.00 %	100.00 %	100.00 %
largeComplexParallel	97.81 %	100.00 %	100.00 %	100.00 %	100.00 %
Exoten-Datensatz	**57.64 %**	**49.00 %**	**54.00 %**	**60.50 %**	**88.50 %**
extraLarge	66.24 %	92.00 %	93.00 %	100.00 %	92.00 %
manyInstances	49.03 %	6.00 %	15.00 %	21.00 %	85.00 %
Durchschnitt	**74.51 %**	**92.64 %**	**93.43 %**	**94.36 %**	**98.29 %**

Tabelle 14: Ergebnisse des Zweiklassenvergleichs auf den PLG-Datensätzen

PLG-Datensätze In Tabelle 14 werden die Ergebnisse des Zweiklassenvergleichs auf den PLG-Datensätzen dargestellt. Abgesehen von den Exoten-Datensätzen können mit Ausnahme eines einzelnen Prozesses, alle Kandidatenpaare korrekt identifiziert werden. Der falsch klassifizierte Prozess ist im *klein/einfach*-Szenario der Basis-Datensätze enthalten.

Die durchschnittlich höchste Genauigkeit wird bei den Parallel-Datensätzen erzielt. Obwohl alle Rausch-Datensätze korrekt klassifiziert werden, wirkt sich die Menge an Rauschen negativ auf die Genauigkeit des Modells aus. Werden die Szenarien jeder Kategorie betrachtet, fällt auf, dass komplexere und größere Prozesse (*largeComplex*) in der Regel eine höhere Genauigkeit aufweisen als kleinere und einfachere (*smallSimple*).

Die schlechtesten Ergebnisse werden auf den Exoten-Datensätzen erzielt. Dennoch kann in 92 Prozent der Fälle das korrekte Kandidatnepaar in den Prozessen des *extraLarge*-

Datensatzes identifiziert werden. Im Gegensatz dazu erzielt das Modell auf den *many-Instances*-Datensätzen schlechte Ergebnisse, da nur in 6 Prozent der Fälle richtig klassifiziert wird. Auffällig auf ist allerdings, dass auf diesem Datensatz dennoch die richtig Aktivitäts-Id zu 85 Prozent erkannt werden kann.

Der Tabelle kann zudem entnommen werden, dass sich die korrekte Klassifikation kaum unterscheidet, wenn anstelle des besten die besten drei Kandidatenpaare betrachtet werden. Dies sorgt nur für eine Steigerung von 1 Prozent.

6.5.3 Dreiklassenvergleich

	Vorhersage positiv	Vorhersage negativ	Vorhersage beide negativ
Tatsächlich positiv	19543 / 33.41 %	19 / 0.03 %	35 / 0.06 %
Tatsächlich negativ	12 / 0.02 %	19363 / 33.10 %	28 / 0.05 %
Tatsächlich beide negativ	158 /0.03 %	156 / 0.03 %	19186 / 32.80 %

Tabelle 15: Wahrheitsmatrix des Dreiklassenvergleichs auf dem Train-Test-Split Datensatz

Train-Test-Split In Tabelle 15 werden die Ergebnisse des Dreiklassenvergleichs auf dem Train-Test-Split Datensatz als Wahrheitstabelle dargestellt. Im Vergleich zu anderen Klassifizierungsstrategien besteht diese aus einer 3×3 Matrix, da zusätzlich das Ergebnis *beide negativ* existiert. Das Modell klassifiziert mit einer Genauigkeit von 99.30 Prozent das korrekte Kandidatenpaar.

Die relevanteste Spalte dieser Matrix ist die *Vorhersage beide negativ*, da diese zu einer Reduktion der Wertung beider Kandidatenpaare führt. Der Tabelle kann entnommen werden, dass nur etwa 1 Prozent der korrekten Kandidatenpaare als *beide negativ* klassifiziert werden.

PLG-Datensätze Die Ergebnisse des Dreiklassenvergleichs sind in Tabelle 16 dargestellt. Auch bei dieser Strategie werden die besten Ergebnisse auf den Parallel-Datensätzen (100 Prozent) erzielt, dicht gefolgt von den Basis-Datensätzen (99 Prozent).

Rauschen hat einen großen Einfluss bei dieser Strategie, da nur 69 Prozent bei 10 Prozent Rauschen und 63 Prozent bei 30 Prozent Rauschen erkannt werden können.

Datensatz	Genauigkeit	Richtig Klassifiziert	Top 3	Richtige Fall-Id	Richtige Aktivitäts-Id
Basis-Datensätze	**77.39 %**	**99.33 %**	**99.33 %**	**99.33 %**	**99.33 %**
smallSimple	52.91 %	98.00 %	98.00 %	98.00 %	98.00 %
smallComplex	84.48 %	100.00 %	100.00 %	100.00 %	100.00 %
largeComplex	94.77 %	100.00 %	100.00 %	100.00 %	100.00 %
Rausch-Datensätze-10	**48.62 %**	**69.16 %**	**98.66 %**	**100.00 %**	**69.16 %**
smallSimpleNoise10	54.07 %	87.00 %	99.00 %	100.00 %	87.00 %
smallComplexNoise10	46.84 %	72.00 %	99.00 %	100.00 %	72.00 %
largeComplexNoise10	44.95 %	48.48 %	97.98 %	100.00 %	48.48 %
Rausch-Datensätze-30	**44.98 %**	**63.83 %**	**98.99 %**	**100.00 %**	**63.83 %**
smallSimpleNoise30	45.75 %	81.00 %	100.00 %	100.00 %	81.00 %
smallComplexNoise30	44.25 %	62.00 %	99.00 %	100.00 %	62.00 %
largeComplexNoise30	44.95 %	48.48 %	97.98 %	100.00 %	48.48 %
Parallel-Datensätze	**83.19 %**	**100.00 %**	**100.00 %**	**100.00 %**	**100.00 %**
smallSimpleParallel	69.75 %	100.00 %	100.00 %	100.00 %	100.00 %
smallComplexParallel	87.79 %	100.00 %	100.00 %	100.00 %	100.00 %
largeComplexParallel	92.03 %	100.00 %	100.00 %	100.00 %	100.00 %
Exoten-Datensätze	**34.37 %**	**45.00 %**	**84.50 %**	**86.50 %**	**45.00 %**
extraLarge	46.89 %	38.00 %	69.00 %	100.00 %	38.00 %
manyInstances	21.85 %	52.00 %	100.00 %	73.00 %	52.00 %
Durchschnitt	**59.38 %**	**77.64 %**	**97.14 %**	**97.93 %**	**77.64 %**

Tabelle 16: Ergebnisse des Dreiklassenvergleichs auf den PLG-Datensätzen

Werden die Szenarien betrachtet, fällt auf, dass in den Basis- und Parallel-Datensätzen große und komplexere Prozesse besser erkannt werden können. In den Rausch-Datensätzen erzielen alle Szenarien ähnliche Ergebnisse.

Für den *manyInstances* Datensatz kann beobachtet werden, dass trotz einer geringen Genauigkeit von 21 Prozent, dennoch 52 Prozent der richtigen Kandidatenpaare klassifiziert werden können. Dies kann damit Begründet werden, dass die Genauigkeit nicht die prognostizierte Wahrscheinlichkeit enthält, mit der zwei falsche Kandidatenpaare als *beide negativ* klassifiziert werden. Diese fließen nicht in die Berechnung ein, damit die Genauigkeit für alle Strategien identisch berechnet wird und somit dieselbe Bedeutung hat. Allerdings zeigt sich anhand des *manyInstances* Datensatzes, dass die negative Beeinflussung zweier falscher Kandidatenpaare einen positiven Einfluss auf die Ergebnisse haben kann.

Dies wird zudem sichtbar, wenn nicht nur das beste, sondern die besten 3 Kandidatenpaare untersucht werden. In diesem Fall verbessern sich die Ergebnisse erheblich. Anstelle von 77 Prozent können nun 97 Prozent aller korrekten Kandidatenpaare richtig identifiziert werden. Selbst in dem *manyInstances*-Datensatz, der eine Herausforderung für andere Strategien darstellt, wird 100 Prozent richtig klassifiziert.

Auffällig ist zudem, dass der Dreiklassenvergleich die Fall-Id zu 98 Prozent richtig prognostiziert, auch wenn generell nur ca. 78 Prozent der Kandidatenpaare richtig identifiziert werden. Die Identifikation der korrekten Fall-Id liegt um 20 Prozent höher als die der korrekten Aktivitäts-Id

6.5.4 Vergleich der Klassifizierungsstrategien

In den vorherigen Unterunterabschnitten werden die individuellen Ergebnisse der Klassifizierungsstrategien detailliert vorgestellt und ihre Besonderheiten beschrieben. Dieser Abschnitt befasst sich mit einem Vergleich der Strategien untereinander.

	Richtig klassifiziert			**Top 3**		
	Direkt	2k-Vergleich	3k-Vergleich	Direkt	2k-Vergleich	3k-Vergleich
Basis-Datensätze	92.00 %	99.67 %	99.33 %	99.33 %	100.00 %	99.33 %
Rausch-Datensätze-30	62.14 %	100.00 %	63.83 %	98.66 %	100.00 %	98.99 %
Parallel-Datensätze	94.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
extraLarge	26.00 %	92.00 %	38.00 %	64.00 %	93.00 %	69.00 %
manyInstances	1.00 %	6.00 %	52.00 %	37.00 %	15.00 %	100.00 %
Durchschnitt	**55.03 %**	**79.53 %**	**70.63 %**	**79.80 %**	**81.60 %**	**93.464 %**

Tabelle 17: Ergebnisse des Dreiklassenvergleich auf den PLG-Datensätzen

In Tabelle 17 werden die Ergebnisse der Direktprognose (*Direkt*), des Zweiklassenvergleichs (*2k-Vergleich*) und des Dreiklassenvergleichs (*3k-Vergleich*) gegenübergestellt. Für den Vergleich werden die Durchschnittswerte der verschiedenen Kategorien verwendet. Jeder für den Vergleich verwendete Datensatz prüft die Modelle auf unterschiedliche Herausforderungen.

Die Kategorie *Rausch-Datensätze-10* wird nicht verwendet, da diese dieselbe Herausforderung wie die *Rausch-Datensätze-30* darstellt, jedoch in einer abgeschwächten Variante. Ebenso werden die Exoten-Datensätze in die einzelnen Datensätze aufgespalten, da diese die Strategien auf drastisch unterschiedliche Herausforderungen testen und die Bildung eines Durchschnitts somit den Informationsgehalt reduziert.

In der letzten Zeile der Tabelle werden die Durchschnittswerte der Kategorien dargestellt, die beschreiben, wie gut die Modelle generell auf unterschiedliche Herausforderungen anwendbar sind.

Die Strategien werden anhand zweier Metriken miteinander verglichen: der prozentualen Häufigkeit, mit der die erste Prognose der Modelle dem korrekten Kandidatenpaar

entspricht, und der prozentualen Häufigkeit, mit der sich dieses unter den ersten drei Prognosen befindet.

Die besten Ergebnisse aller Klassifizierungsstrategien werden auf den Parallel-Datensätzen erzielt. Dies entspricht nicht den Erwartungen, da angenommen wird, dass Nebenläufigkeit zu mehr Mustern mit geringer Auftrittswahrscheinlichkeit führt und somit eine größere Herausforderung für diese Modelle darstellt. Eine mögliche Erklärung dieses Phänomens könnte sein, dass die Attribute in zwei Verhaltensweisen aufgeteilt werden können. Entweder entstehen Muster mit sehr hoher Auftrittswahrscheinlichkeit, bei denen die Attribute in jeder Prozessinstanz denselben oder einen ähnlichen Wertverlauf annehmen, oder es entstehen Muster mit sehr geringer Auftrittswahrscheinlichkeit, da die Attribute für jede Prozessinstanz zufällige Werte erzeugen. Die Modelle könnten daher Muster mit moderater Auftrittswahrscheinlichkeit als Indiz für die korrekte Fall- und Aktivitäts-Id werten. Mit diesem Verhalten könnte erklärt werden, weshalb die Parallel-Datensätze bessere Ergebnisse erzielen, da Nebenläufigkeit zu einer höheren Anzahl an Mustern führt und somit die Auftrittswahrscheinlichkeit aller Muster sinkt.

Direktvergleich In nahezu allen Metriken werden durch den Direktvergleich schlechtere Ergebnisse im Vergleich zu den anderen Strategien erzielt. Besonders bei der direkten Prognose des korrekten Kandidatenpaars zeigt sich eine große Differenz. Rauschen in den Eventlogs und die Exoten-Datensätze beeinflussen die Resultate deutlich stärker als die Vergleichsstrategien. Werden jedoch die besten 3 Prognosen des Modells betrachtet, verbessern sich die Ergebnisse erheblich. Obwohl diese immer noch hinter den anderen Strategien liegen, beträgt der Unterschied nur noch wenige Prozentpunkte. Ein großer Vorteil der Direktprognose ist die geringe Laufzeitkomplexität. Die Berechnung der Ergebnisse auf allen Testdatensätzen benötigt weniger als 10 Sekunden. Auf demselben System benötigen die Vergleichsstrategien mehr als eine Stunde für die Verarbeitung aller Testdatensätze. Für Eventlogs, die eine große Anzahl an Attributen enthalten, bietet die geringe Laufzeitkomplexität daher einen erheblichen Vorteil gegenüber den Vergleichsstrategien.

Zweiklassenvergleich Mit dem Zweiklassenvergleich werden die besten Ergebnisse bei der direkten Prognose des korrekten Kandidatenpaars erzielt. Diese Strategie ist besonders resilient gegenüber Rauschen. Nur in dem Zweiklassenvergleich können alle Datensätze, in denen Rauschen enthalten ist, richtig klassifiziert werden. Im Vergleich zu

den anderen Strategien können besonders gute Ergebnisse auch auf den großen Prozessen (*extraLarge*) nachgewiesen werden.

Allerdings wird die Annahme bestätigt, dass einzelne Fehlentscheidungen des Modells drastische Auswirkungen auf die Ergebnisse haben können. Wenn nicht nur das beste, sondern die besten drei Prognosen betrachtet werden, verbessern sich die Resultate nur marginal (2 Prozent). Wenn das prognostizierte Kandidatenpaar nicht korrekt ist, kann somit keine Aussage über weitere potenzielle Kandidaten für die Fall- und Aktivitäts-Id getroffen werden. Dies ist bei den anderen Strategien nicht der Fall, da dort das als zweites prognostizierte Kandidatenpaar verwendet werden kann.

Dreiklassenvergleich Die Resultate des Dreiklassenvergleichs liegen zwischen denen der Direktprognose und des Zweiklassenvergleichs bei der direkten Prognose des korrekten Kandidatenpaars. Besonders in den Rausch-Datensätzen und den extra großen Prozessen (*extraLarge*) wird eine große Differenz zum Zweiklassenvergleich sichtbar. Allerdings zeigt der Dreiklassenvergleich als einzige Strategie die Fähigkeit, die Fall- und Aktivitäts-Id bei Prozessen mit vielen Prozessinstanzen zu identifizieren. Dies ist besonders hervorzuheben, da diese beim Zweiklassenvergleich und der Direktprognose nur bei 6 bzw. 1 Prozent liegen.

Die Besonderheit des Dreiklassenvergleichs im Vergleich zum Zweiklassenvergleich liegt darin, dass sich zwei falsche Kandidatenpaare gegenseitig negativ beeinflussen können. Es wird angenommen, dass durch diese gegenseitige Beeinflussung die Auswirkungen einzelner Fehlentscheidungen der Modelle weniger stark ins Gewicht fallen. Die Ergebnisse belegen diese Annahme, da der Dreiklassenvergleich die besten Ergebnisse aller Strategien erzielt, wenn anstelle des ersten Prognose, die besten drei Prognosen betrachtet werden. In den Basis- und Rausch-Datensätzen ist der Dreiklassenvergleich nur marginal schlechter als der Zweiklassenvergleich. Allerdings werden auf den Exoten-Datensätzen deutlich bessere Ergebnisse erzielt. Bei den anderen Strategien befindet sich auf diesen Datensätzen im Durchschnitt nur in weniger als 50 Prozent der Fälle das korrekte Kandidatenpaar unter den besten drei Prognosen. Der Dreiklassenvergleich erzielt dagegen 85 Prozent.

6.5.5 Fazit

Mit den Ergebnissen der Evaluation wird belegt, dass alle drei vorgestellten Klassifizierungsstrategien in der Lage sind, die korrekte Fall- und Aktivitäts-Id in unlabeled Eventlogs zu identifizieren. Für jede Strategie konnten Vor- und Nachteile ermittelt werden:

Für die Direktprognose werden die schlechtesten Ergebnisse festgestellt. Allerdings weist diese Strategie eine geringe Laufzeitkomplexität auf und ist in der Lage, die Kandidatenpaare nach ihrer Wahrscheinlichkeit zu sortieren. Dadurch kann diese Strategie als Vorabfilter für Kandidatenpaare verwendet werden, insbesondere in Szenarien, in denen die Laufzeitkomplexität der Vergleichsstrategien auf allen Kandidatenpaaren zu hoch ist.

Mit dem Zweiklassenvergleich können zwar die besten Ergebnisse erzielt werden, jedoch sind Fehlentscheidungen des Modells schwerwiegender. Wenn Prozesse falsch klassifiziert werden, kann keine Aussage über die Wahrscheinlichkeit getroffen werden, mit der die verbleibenden Kandidatenpaare der korrekten Fall- und Aktivitäts-Id entsprechen. In Absatz 1.1 wird beschrieben, wie neue Perspektiven auf Geschäftsprozesse erlangt werden können, wenn Prozess-Mining auf Feldern durchgeführt wird, die nicht der Fall- und Aktivitäts-Id entsprechen. Dies setzt eine Gewichtung der Kandidatenpaare voraus, die mit dem Zweiklassenvergleich nicht erzielt werden kann.

Der Dreiklassenvergleich liefert gute Ergebnisse und ermöglicht eine Gewichtung aller Kandidatenpaare basierend auf ihrer Wahrscheinlichkeit, mit dem sie dem korrekten Kandidatenpar entsprechen. Zudem erzielt diese Strategie die besten Resultate auf Datensätzen, die stark von den Trainingsdaten abweichen (Exoten-Datensätze). Dies führt zu der Annahme, dass der Dreiklassenvergleich auch auf realen Eventlogs bessere Ergebnisse liefern wird, da reale Eventlogs stark in Größe, Komplexität und der Anzahl an Prozessinstanzen variieren können. Die Klassifizierung von drei Zuständen ist zudem deutlich komplexer als die des Zweiklassenvergleichs. Es besteht die Möglichkeit, dass der Dreiklassenvergleich durch die Hyperparameter aus Unterabschnitt 6.2 limitiert wird und ein komplexeres Modell zu einer Leistungssteigerung führen könnte. Für den Prototypen wird daher der Dreiklassenvergleich als Standardimplementierung verwendet, da dieser die größten Potenziale aufweist.

7 Überprüfung der Annahmen des Konzeptes

In Unterabschnitt 3.4 wurden 3 Hypothesen aufgestellt, auf denen das Konzept zur Identifikation der Fall- und Aktivitäts-Id basiert. Abschließend ist zu prüfen, inwiefern diese durch die Implementierung belegt werden.

Annahme 1 *Die Fall- und Aktivitäts-Ids können anhand charakteristischer Merkmale der Muster von anderen Attributen im Eventlog unterschieden werden.*

Annahme 2 *Ein intelligentes System ist in der Lage diese Charakteristika für Geschäftsprozesse unterschiedlicher Größe und Komplexität zu erlernen.*

In Unterunterabschnitt 4.3.4 werden Kennzahlen vorgestellt, die die charakteristischen Merkmale der Fall- und Aktivitäts-Id quantifizieren. Diese dienen als Datengrundlage für das intelligente System.

In Unterabschnitt 6.4 werden mehrere Klassifizierungsstrategien vorgestellt, die anhand eines intelligenten Systems und diesen Kennzahlen die Identifikation der Fall- und Aktivitäts-Id ermöglichen. Durch die anschließende Evaluation wird gezeigt, dass die vorgestellten Strategien in der Lage sind, die korrekte Fall- und Aktivitäts-Id zu identifizieren. Somit werden beide Annahmen belegt.

Annahme 3 *Obwohl Rauschen, Nebenläufigkeit und Zyklen die Musterbildung beeinflussen können, schränken sie die Bildung von Mustern und damit die Identifikation der Fall- und Aktivitäts-Ids nur bedingt ein.*

Für die Evaluation werden Prozesse verwendet, die unterschiedliche Szenarien abbilden. Darunter befinden sich auch Eventlogs, die Rauschen enthalten, sowie Prozesse mit Nebenläufigkeit und Zyklen (siehe Unterunterabschnitt 6.3.2). Aus den Ergebnissen kann entnommen werden, dass auch auf diesen Datensätzen die korrekte Fall- und Aktivitäts-Id identifiziert werden kann und sich diese Szenarien nur leicht auf die Genauigkeit der Klassifikation auswirken. Somit gilt diese Annahme ebenfalls als belegt.

8 Fazit und Ausblickt

8.1 Zusammenfassung und Fazit

In dieser Arbeit wird die zentrale Forschungsfrage gestellt:

Wie können Felder für Fall- und Aktivitäts-Id in unlabeled Eventlogs automatisiert identifiziert werden?

Zur Beantwortung dieser Frage wird ein Konzept vorgestellt, das die automatisierte Identifikation der Fall- und Aktivitäts-Ids in unlabeled Eventlogs ermöglicht. Das Konzept nutzt wiederkehrende Muster, um charakteristische Merkmale für Fall- und Aktivitäts-Ids zu bestimmen. Diese Merkmale werden durch Kennzahlen quantifiziert. Es wird angenommen, dass diese Kennzahlen von einem intelligenten System erlernt werden können, um die Fall- und Aktivitäts-Ids zu prognostizieren.

Um diese Annahmen zu bestätigen, wird das Konzept in Form eines Prototyps umgesetzt. Für die Prognosen der korrekten Fall- und Aktivitäts-Id werden verschiedene Klassifizierungsstrategien vorgestellt. Diese Strategien werden umfassend auf Prozessen unterschiedlicher Größe und Komplexität evaluiert und zeigen die Fähigkeit, die korrekten Felder für Fall- und Aktivitäts-Ids zuverlässig zu identifizieren. Somit wird der Beleg erbracht, dass die Forschungsfrage mit dem vorgestellten Konzept beantwortet werden kann.

Da eine unzureichende Anzahl an frei zugänglichen Daten für das Anlernen des intelligenten Systems existiert, sind als Nebenprodukt dieser Arbeit eine Prozess-Engine und ein Prozess-Generator entstanden. Diese Systeme sind in der Lage, zufällige, unvoreingenommene Geschäftsprozesse unterschiedlicher Größe und Komplexität zu erzeugen und zu simulieren.

8.2 Ausblick

Das vorgestellte Konzept stellt einen neuen Ansatz zur Identifikation der Fall- und Aktivitäts-Id in unlabeled Eventlogs dar. Dementsprechend können zahlreiche weiterführende Untersuchungen und Erweiterungen für das Konzept abgeleitet werden. Im Folgenden werden einige vielversprechende Untersuchungen und Erweiterungen vorgestellt, deren Umsetzungen im zeitlichen Rahmen dieser Arbeit nicht möglich waren:

Evaluation mit realen Eventlogs Der Prototyp verwendet bisher nur synthetische Eventlogs für das Training und die Evaluation. Zwar werden große Bemühungen unternommen, die synthetischen Daten möglichst praxisnah zu gestalten, jedoch steht der Beleg noch aus, ob das Konzept und der Prototyp auch auf realen Datensätzen erfolgreich angewendet werden kann. Mit einem umfassenden Datensatz bestehend aus realen Eventlogs kann dieser Beleg erbracht werden.

Erweiterung des Trainingsbestands Der Trainingsdatensatz umfasst Prozesse mit einer vordefinierten Größe und Komplexität. Die Evaluation zeigt, dass Prozesse, die stark von diesen definierten Bereichen abweichen, die angelernten Modelle vor große Herausforderungen stellen. Daher ist zu untersuchen, inwiefern sich dies verändert, wenn neue Szenarien in den Trainingsdatensatz aufgenommen werden. Dies könnte zu Modellen führen, die auf ein größeres Spektrum an Prozessen anwendbar sind und somit auch auf realen Eventlogs in der Praxis bessere Ergebnisse erzielen.

Weitere Kennzahlen Um die Charakteristika der Fall- und Aktivitäts-Id zu quantifizieren, werden 12 Kennzahlen aus den Mustern im Eventlog errechnet. Weitere Kennzahlen können abgeleitet werden, um die Charakteristika der Fall- und Aktivitäts-Id genauer abzubilden. Da diese als Datengrundlage für die Klassifikation des intelligenten Systems dienen, beeinflusst der Informationsgehalt der Kennzahlen maßgeblich die Genauigkeit der Modelle.

Komplexere Modelle Für die Evaluation werden nur Random-Forest-Modelle mit begrenzter Komplexität getestet, um den Trainingsprozess zu verkürzen. Gerade bei dem Dreiklassenvergleich besteht die Möglichkeit, dass die Modelle in ihrer Funktionalität

eingeschränkt werden. Es ist zu untersuchen, ob komplexere Modelle oder andere Technologien des maschinellen Lernens bessere Ergebnisse erzielen

Literatur

[1] Aalst, Wil ; Medeiros, Ana ; Weijters, A.: Genetic Process Mining, 06 2005, S. 48–69. – ISBN 978-3-540-26301-2

[2] Ahmad, Muhammad W. ; Mourshed, Monjur ; Rezgui, Yacine: Trees vs Neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption. In: *Energy and buildings* 147 (2017), S. 77–89

[3] Ahmed, Shahzad ; Ali, M U. ; Ferzund, Javed ; Sarwar, Muhammad A. ; Rehman, Abbas ; Mehmood, Atif: Modern data formats for big bioinformatics data analytics. In: *arXiv preprint arXiv:1707.05364* (2017)

[4] Alpaydin, Ethem: *Introduction to machine learning*. MIT press, 2020

[5] Alpaydin, Ethem: *Machine learning*. MIT press, 2021

[6] Baier, Thomas ; Di Ciccio, Claudio ; Mendling, Jan ; Weske, Mathias: Matching events and activities by integrating behavioral aspects and label analysis. In: *Software & Systems Modeling* 17 (2018), Nr. 2, S. 573–598

[7] Bala, Saimir ; Mendling, Jan ; Schimak, Martin ; Queteschiner, Peter: Case and activity identification for mining process models from middleware. In: *IFIP Working Conference on The Practice of Enterprise Modeling* Springer (Veranst.), 2018, S. 86–102

[8] Bayomie, Dina ; Awad, Ahmed ; Ezat, Ehab: Correlating unlabeled events from cyclic business processes execution. In: *International Conference on Advanced Information Systems Engineering* Springer (Veranst.), 2016, S. 274–289

[9] Berti, Alessandro ; Zelst, Sebastiaan van ; Schuster, Daniel: PM4Py: A process mining library for Python. In: *Software Impacts* 17 (2023), S. 100556. – URL https://www.sciencedirect.com/science/article/pii/S2665963823000933. – ISSN 2665-9638

[10] Breiman, Leo: Random forests. In: *Machine learning* 45 (2001), S. 5–32

[11] Burattin, Andrea: Plg2: Multiperspective process randomization with online and offline simulations. In: *BPM (Demos)* Citeseer (Veranst.), 2016, S. 1–6

[12] Burattin, Andrea ; Sperduti, Alessandro: PLG: A framework for the generation of business process models and their execution logs. In: *Business Process Management Workshops: BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13-15, 2010, Revised Selected Papers 8* Springer (Veranst.), 2011, S. 214–219

[13] Cheng, Hsin-Jung ; Kumar, Akhil: Process mining on noisy logs - Can log sanitization help to improve performance? In: *Decision Support Systems* 79 (2015), 11, S. 138–149

[14] Fernandez-Delgado, Manuel ; Cernadas, E. ; Barro, S. ; Amorim, Dinani: Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? In: *Journal of Machine Learning Research* 15 (2014), 10, S. 3133–3181

[15] Ferreira, Diogo R. ; Gillblad, Daniel: Discovering process models from unlabelled event logs. In: *International Conference on Business Process Management* Springer (Veranst.), 2009, S. 143–158

[16] Kecht, Christoph ; Egger, Andreas ; Kratsch, Wolfgang ; Röglinger, Maximilian: Event Log Construction from Customer Service Conversations Using Natural Language Inference. In: *2021 3rd International Conference on Process Mining (ICPM)* IEEE (Veranst.), 2021, S. 144–151

[17] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[18] Pourmirza, Shaya ; Dijkman, Remco ; Grefen, Paul: Correlation mining: mining process orchestrations without case identifiers. In: *International Conference on Service-Oriented Computing* Springer (Veranst.), 2015, S. 237–252

[19] Probst, Philipp ; Wright, Marvin N. ; Boulesteix, Anne-Laure: Hyperparameters and tuning strategies for random forest. In: *Wiley Interdisciplinary Reviews: data mining and knowledge discovery* 9 (2019), Nr. 3, S. e1301

[20] Raff, Edward: JSAT: Java Statistical Analysis Tool, a Library for Machine Learning. In: *Journal of Machine Learning Research* 18 (2017), Nr. 23, S. 1–5. – URL http://jmlr.org/papers/v18/16-131.html

[21] Sahu, Muktikanta ; Nayak, Gopal K. ; Nayak, Rupaj K.: Process Model Discovery from Unlabeled Event Logs by Using Non-Overlapping Sequential Distinct Event Patterns.

[22] team, The pandas development: *pandas-dev/pandas: Pandas*. Februar 2020. – URL https://doi.org/10.5281/zenodo.3509134

[23] Turner, Ryan ; Eriksson, David ; McCourt, Michael ; Kiili, Juha ; Laaksonen, Eero ; Xu, Zhen ; Guyon, Isabelle: Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In: *NeurIPS 2020 Competition and Demonstration Track* PMLR (Veranst.), 2021, S. 3–26

[24] Van Der Aalst, Wil ; Adriansyah, Arya ; De Medeiros, Ana Karla A. ; Arcieri, Franco ; Baier, Thomas ; Blickle, Tobias ; Bose, Jagadeesh C. ; Van Den Brand, Peter ; Brandtjen, Ronald ; Buijs, Joos u. a.: Process mining manifesto. In: *Business Process Management Workshops: BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I 9* Springer (Veranst.), 2012

[25] Van Rossum, Guido ; Drake Jr, Fred L.: *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995

[26] Zhu, Xiaochen ; Song, Shaoxu ; Lian, Xiang ; Wang, Jianmin ; Zou, Lei: Matching heterogeneous event data. In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, S. 1211–1222

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original