Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 208

Mining und Visualisierung von Prozessen in einem komplexen Softwaresystem

Fabian Gajek

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Thomas Ertl
Betreuer/in:	Dipl.-Inf. Dominik Herr, Dr. Steffen Lohmann
Beginn am:	22. April 2015
Beendet am:	22. Oktober 2015
CR-Nummer:	H.5.2, K.6.3

Kurzfassung

Komplexe Software-Anwendungssysteme haben oft eine Vielzahl an Prozessen, welche mit sehr vielen Teilkomponenten interagieren. Da bei einer agilen Entwicklung häufig die bestehenden Prozesse nicht ausreichend dokumentiert oder Änderungen nicht übertragen werden, wird die Entwicklung und Wartung deutlich erschwert. Process-Mining bietet Techniken, die Prozessmodelle mit den reellen Prozessen zu synchronisieren. Diese Arbeit untersucht die Anwendung von Business-Process-Mining auf komplexen Softwaresystemen anhand von Daten aus der Automobilindustrie und die visuelle Aufbereitung der erstellten Modelle, mit dem Ziel die Analyse der Prozesse zu erleichtern. Dazu werden die Darstellung der Prozessmodelle mithilfe eines kräftebasierten Layouts optimiert und die dem Modell zugrunde liegenden Abläufe in die Visualisierung integriert.

Abstract

Complex software systems often contain a lot of processes that communicate between many different subsystems. Agile development leads to under-documented processes or changes which are not synchronized between implementation and model. This complicates the development and maintenance of the system. Process Mining offers techniques to synchronize the real world with the model. This thesis investigates the application of Business Process Mining to complex software systems and the subsequent visual presentation of the mined models using example data from the automotive industry. The visualization of the model is optimized using a force directed layout and augmented through the integration of the underlying traces.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Verzeichnis der Algorithmen

1. Einleitung und Aufgabenstellung

Komplexe Anwendungssysteme bestehen aus vielen Komponenten, welche auf verschiedenen Knoten ausgeführt werden und über Schnittstellen miteinander kommunizieren. Anfragen betreten das System an einer Komponente und werden danach mit großem Zusammenspiel der Subsysteme abgearbeitet. Je nach Größe des Systems ist es schwierig, den Überblick über den Ablauf von Verarbeitungen zu haben. Deswegen sind die entsprechenden Abläufe meist genau spezifiziert und dokumentiert. Jedoch ist der Idealzustand von einem vollständig spezifizierten System und einer genau der Spezifikation entsprechenden Implementierung in der Praxis oft nicht vorzufinden. Zum einen werden Abläufe von Verarbeitungen in der Implementierung geändert, gerade heutzutage durch die weite Verbreitung von agiler Softwareentwicklung, und dabei vergessen die entsprechende Spezifikation an die Änderungen anzupassen. Des weiteren kann es passieren, dass neue Abläufe eingeführt werden und nicht ausreichend oder gar nicht dokumentiert werden. Es existiert eine Vielzahl von weiteren Ursachen, z.B. Fehler in der Implementierung, warum die Abläufe im fertigen System nicht der Spezifikation entsprechen.

Business-Process-Mining verspricht, dieses Problem zu lösen. Es zielt im Allgemeinen darauf ab, die Ausführung von Businessprozessen auf das Ausführungsmodell zurückzuführen. Dafür bietet es Algorithmen, um Prozessabläufe auf Übereinstimmung mit dem Prozessmodell zu validieren, Prozessmodelle nur aus dokumentierten Abläufen zu erstellen und Techniken um existierende Modelle anhand in der Praxis gemessener Daten zu verbessern. Da diese Konzepte auch auf Abläufe in Softwaresystemen angewendet werden können, bietet Process-Mining eine Möglichkeit, um oben genannte Probleme zu verbessern. Die Dokumentation kann ohne großen Aufwand aktuell gehalten werden durch Algorithmen, die Ablaufmodelle aus den Abläufen erstellen, oder durch Systeme, welche Abweichung von dem spezifizierten Kontrollfluss erkennt.

Diese Arbeit beschäftigt sich mit dem Erstellen von Prozessmodellen aus Logeinträgen großer Anwendungssysteme. Die anderen Techniken des Process-Mining werden nicht untersucht. Dazu werden reale Daten eines Backendsystemes eines deutschen Autoherstellers verwendet, um den Ansatz zu testen.

Da die erzeugten Modelle von Menschen verstanden und interpretiert werden sollen, wird untersucht, ob eine geeignete Visualisierung die Analyse unterstützt. Zusätzlich dazu werden die Abläufe aus den Logdaten in die Darstellung integriert, um die Plausibilität und das Verständnis des erzeugten Modelles zu unterstützen.

1.1. Situation

In der Automobilindustrie werden immer mehr computergesteuerte Dienste im Auto angeboten. Dies geht von elektronischen Fahrt unterstützenden Systemen bis hin zu Unterhaltungselektronik, mit der man im Auto im Internet surfen kann. Aber es existiert auch eine Klasse von Diensten, die über die Elektronik des Autos hinaus mit Backendsystemen beim Hersteller kommunizieren. Dabei handelt es sich um Dienste, welche die Fernsteuerung des Autos durch den Nutzer ermöglichen oder Information über den Zustand des Autos, etwa der Benzinstand, bereitstellen. Für diese Dienste werden komplexe Softwaresysteme bei den Herstellern aufgebaut in denen zum Teil größere Prozesse ablaufen. Dies hat seine Gründe auch in der Vielfalt der Fahrzeugausstattungen und der Forderung nach Datenschutz und Informationssicherheit, die bei den Bewegungsdaten eines Autos eine große Bedeutung haben. Diese Arbeit soll untersuchen, wie die Process-Discovery die Entwicklung und Wartung eines solchen Systemes verbessern kann.

1.2. Aufgabe

Das Ziel der Arbeit ist die Verbesserung der Analyse von Prozessen in komplexen Softwaresystemen wie das oben beschriebene. Dazu sollen geeignete Prozessmodelle aus den vorhandenen Logdaten des Softwaresystems extrahiert werden. Auch soll eine passende Visualisierung entwickelt werden und um Interaktionsmöglichkeiten erweitert werden, um eine Analyse der Prozesse weiter zu verbessern. Das Ziel der Arbeit ist ein System, mit dem die Erstellung von Prozessmodellen möglich ist, welche die Analyse von Abläufen vereinfachen. Dabei sollen auch tatsächliche Prozesse mit gegeben Modellen verglichen werden, um Abweichung und Probleme zu erkennen.

Diese Arbeit verbindet das Process-Mining oder genauer die Process-Discovery mit Techniken der Visualisierung. Gleichzeitig untersucht sie die Anwendbarkeit von Techniken der Process-Discovery auf Abläufe in Softwaresystemen. Dazu werden reale Daten aus der Automobilindustrie verwendet und Experten des analysierten zu einer Studie herangezogen. Die folgenden Arbeitsschritte sind Aufgabe der Arbeit:

1. Auseinandersetzung mit den Logdaten und verwandten Arbeiten
2. Konzeption und Entwicklung eines prototypischen Systems
3. Anwendung des entwickelten Systems auf die vorhandenen Logdaten
4. Evaluation des entwickelten Ansatzes mittels einer Expertenbefragung
5. Auswertung und Diskussion der Evaluationsergebnisse

1.3. Lösungsansatz

Die Entwicklung des Systems ist aufgeteilt nach den Komponenten, welche die einzelnen Verarbeitungsschritte übernehmen sollen.

1. Die gegeben Logdaten müssen analysiert werden und daraus ein Konzept entwickelt werden, das es ermöglicht aus den existierenden Logeinträgen, in natürlicher Sprache, eine strukturierte XES-Datei (Unterabschnitt 2.5.3) für einen bestimmten Prozess erstellt werden kann. Aufbauend auf diesem Konzept wird dieser Ablauf in einer Komponente des Systems implementiert, welche diese Verarbeitung automatisiert mit möglichst geringer Nutzerinteraktion übernimmt.
2. Der nächste Schritt der Verarbeitung ist die Erstellung eines Prozessmodells. Daher sollte das Konzept entworfen werden, wie diese aus einem Log erstellt werden kann. Dabei ist zu überlegen, welche vorhandenen Algorithmen aus dem Process-Mining verwendet werden sollen, wie diese implementiert und gegebenenfalls konfiguriert werden. Daraus sollte eine Komponente entstehen, welche als Eingabe Logdaten (möglicherweise auch die Wahl eines Algorithmus) und als Ausgabe ein Prozess Modell liefert. Es sollte möglich sein, das Modell sowohl als Petri-Netz als auch als BPMN zu bekommen.
3. Nach der Erstellung der Modelle müssen diese visualisiert werden. Dazu ist es nötig zu überlegen wie Petri-Netze und BPMN am besten visualisiert werden können und wie diese noch um zusätzliche Interaktion oder Visualisierung erweitert werden können, um die Analyse zu verbessern. Dies sollte in einem Frontend zusammengefasst werden, das nicht nur die Diagramme visualisiert, sondern auch die anderen Komponenten mit möglich wenig Aufwand steuert.

1.4. Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen: Hier werden die Grundlagen dieser Arbeit beschrieben. Diese reichen von der Businessprozessmodellierung bis hin zur Visualisierung von Graphen.

Kapitel 3 – Konzept: Beschreibung des entwickelten Konzeptes für das prototypische System.

Kapitel 4 – Implementierung: Dieses Kapitel beschreibt die Implementierung des prototypischen Systems.

Kapitel 5 – Evaluation durch Expertenstudie: Die Evaluation des erstellten Systems mittels einer Expertenstudie.

Kapitel 6 – Fazit und Ausblick: Dieses Kapitel fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

2. Grundlagen

Dieses Kapitel beschreibt die Grundlagen, die in dieser Arbeit verwendet werden. Angefangen wird mit einer Einführung in die Prozessmodellierung (Abschnitt 2.1). Dabei liegt der Schwerpunkt auf den Petri-Netzen (Abschnitt 2.2). Darauf folgt das Business Process Model and Notation (kurz BPMN) in Abschnitt 2.3. Der Abschnitt über Information Retrieval (Abschnitt 2.4) gibt eine kurze Einführung, wie Dokumente durchsuchbar gemacht werden können. Folgend beschreibt ein großer Teil das Process Mining (Abschnitt 2.5) und dessen Teilgebiet der Process Discovery (Abschnitt 2.6). Schlussendlich folgt die Visualisierung von Graphen (Abschnitt 2.7), die nötig für die Darstellung der Prozessmodelle ist.

2.1. Prozessmodellierung

Große Unternehmen nutzen Prozessmodellierungen, um ihre Vielzahl von großen Prozessen zu verwalten. Nur so ist es ihnen möglich zum Beispiel auf personelle Änderungen zu reagieren oder Schwachstellen in den Prozessen zu finden. In der Managementebene wird viel mit Prozessflussmodellen gearbeitet, da diese möglicherweise nicht die praktische Anwendung der Prozesse vor Ort gesehen haben. Auch ein Konformitätscheck, die Überprüfung ob sich einzelnen Personen auch an die vorgeben Prozesse halten um die geplanten Anforderungen zu erfüllen, ist nur mit klar definierten Prozessen möglich. Oft werden diese zum Teil informell, etwa mit PowerPoint-Diagrammen spezifiziert. Es werden aber auch immer mehr formale Sprachen, z. B. BPMN, genutzt um die Prozesse zu beschreiben. Durch die klare und formale Definition der Bedeutung dieser Sprachen können Modelle, welche mit den formalen Notationen erstellt wurden, besser analysiert und sogar simuliert werden. Dadurch ergeben sich viele Erkenntnisse, die ohne (formale) Modellierung nur schwer möglich wären.

Dabei existiert eine Vielzahl von formalen Notationen um Prozesse zu beschreiben. Diese Arbeit wird nur Petri-Netze und BPMN verwenden. Dies stellt keine Einschränkung an den Gehalt gewonnener Erkenntnisse dar. Da andere existierende Notationen auf den gleichen Konzepten basieren und zum Teil sogar die gleiche Aussagekraft besitzen, können Erkenntnisse direkt übernommen werden und Notationen ineinander umgewandelt werden.

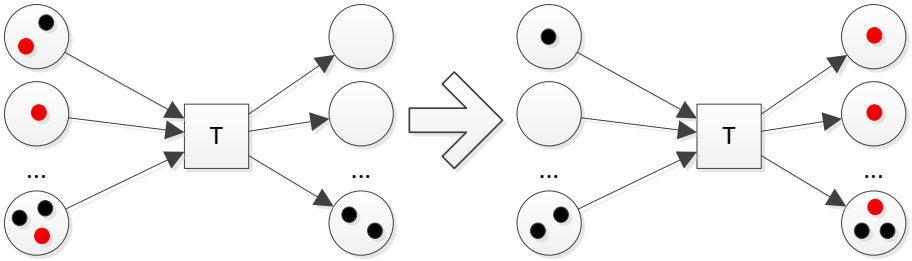


Abbildung 2.1.: Bei dem Feuern einer Transition wird aus jedem Vorgänger eine Marke genommen und in jeden Nachfolger eine Marke gelegt.

2.2. Petri-Netze

Carl Adam Petri begründete die Theorie der Petri-Netze mit seiner Dissertation im Jahre 1962 [Pet62]. Petri-Netze waren eines der ersten Modelle, welche die Möglichkeit besaßen, nebenläufige Abläufe zu modellieren. Viele grundlegende Ideen lassen sich in modernen Notationen, wie BPMN oder Aktivitätsdiagrammen der Unified Modeling Language (UML), wiederfinden. Auch wurden Petri-Netze um verschiedene Arten von Marken semantisch erweitert, um dem Leser die Bedeutung besser verdeutlichen zu können. Bei diesen sogenannten gefärbten Petri-Netzen werden die Marken durch Symbole ausgetauscht, womit eine Marke eine verständliche Bedeutung bekommt.

2.2.1. Grafische Repräsentation

Grafisch lassen sich Petri-Netze als bipartite Graphen repräsentieren. Dabei existieren zwei Arten von Knoten, Stellen, welche als Kreise dargestellt sind, und Transitionen, welche durch Rechtecke repräsentiert werden. Marken (dargestellt durch ausgefüllte Kreise) werden von den Stellen beinhaltet und definieren den Zustand des Netzes.
Transitionen repräsentieren Ereignisse welche in dem modellierten System eintreten können. Damit das Ereignis einer Transition ausgeführt werden kann, müssen alle Vorgänger mindestens eine Marke besitzen. Die Vorgänger einer Transition sind alle Stellen, von denen eine Kante zu der Transition führt. Beim Eintretens des Ereignisses wird aus allen Vorgängern eine Marke genommen und allen Nachfolgern eine Marke hinzugefügt. Nachfolger sind dementsprechend die Stellen, zu denen eine Kante von der Transition führt. Dieser Vorgang des Eintreten eines Ereignisses wird „Feuern“ einer Transition genannt.
Die Stellen grenzen die möglichen Zustände ein, denn die Marken in den entsprechenden Stellen definieren den Zustand. In einer Stelle können sich im Allgemeinen beliebig viele Marken befinden. Um die Notation der Petri-Netze näher zu bringen, folgen einige Beispiele, welche Standardprobleme der nebenläufigen Programmierung modellieren:

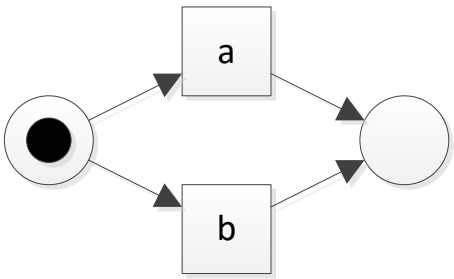


Abbildung 2.2.: Einfaches Petri-Netz, das die exklusive Ausführung von a oder b modelliert.

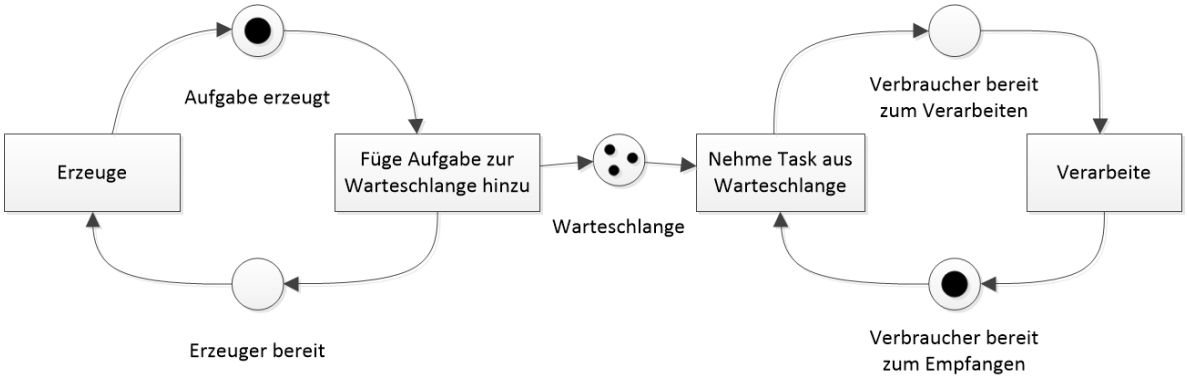


Abbildung 2.3.: Petri-Netz, welches das Erzeuger-Verbraucher-Problem modelliert

Abbildung 2.2 ist ein einfaches Beispiel mit zwei möglichen Abläufen. Dieses Petri-Netz modelliert die exklusive Ausführung von a oder b. Dabei ist die Abbildung in ihrem Ausgangszustand gegeben. Im Petri-Netz aus Abbildung 2.2 wird durch das „Feuern“ einer der Transitionen die Marke aus der linken in die rechte Stelle bewegt. Danach ist keine Transition mehr ausführbar, da sich in allen Stellen der eingehenden Kanten eine Marke befinden muss. Somit modelliert das Netz ein System in dem entweder a oder b einmal ausgeführt werden kann.

Abbildung 2.3 zeigt ein Standardproblem der nebenläufigen Ausführung von Prozessen. Das Erzeuger-Verbraucher-Problem. Dabei geht es im Allgemeinen darum, dass eine oder mehrere Entitäten, die Produzenten, Aufgaben erzeugen, die von einer oder mehreren Entitäten, den Verbrauchern, abgearbeitet werden. In der hier dargestellten Modellierung mit einem Erzeuger und einem Verbraucher, kann der Erzeuger beliebig viele Marken in der Warteschlange erzeugen, da sich die Marke im Modell immer im Kreis mit den Stellen „Aufgabe erzeugt“ und „Erzeuger bereit“ bewegt. Bei jeder Runde wird eine Marke in die Warteschlange gelegt. Die Marken in der zentralen Stelle repräsentieren die Aufträge, welche abgearbeitet werden sollen. Der Verbraucher kann nur dann arbeiten, wenn sich mindestens eine Marke in der

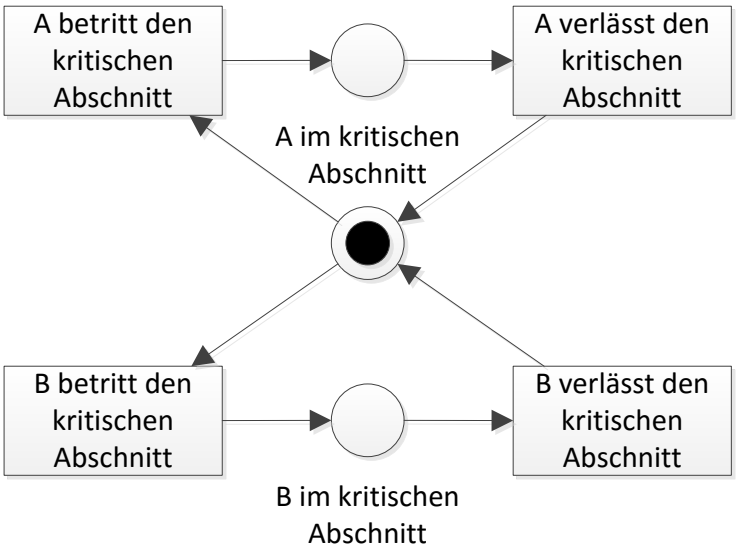


Abbildung 2.4.: Petri-Netz, welches den kritischen Abschnitt modelliert.

Warteschlange sich befindet. Dieses grundlegende Verhalten lässt sich sehr anschaulich mit Petri-Netzen modellieren.

Auch Abbildung 2.4 zeigt die Darstellung eines Problems aus der nebenläufigen Programmierung: das Problem des kritischen Abschnittes. Dabei geht es darum einen bestimmten Codeabschnitt so zu schützen, dass er nur von einem Prozess gleichzeitig betreten werden kann. Das gegebene Modell verdeutlicht dies für zwei „Akteure“, A und B. Dabei existieren drei Zustände: A, B oder niemand befindet sich mit der Ausführung im kritischen Abschnitt. Dabei ist auch vor allem sichergestellt, dass A nur den kritischen Abschnitt betreten kann, wenn B sich nicht darin befindet und umgekehrt.

Weitere Beispiele und Erklärungen sind in [Rei85] zu finden.

2.2.2. Mathematische Definition und Notation

Die mathematischen Definitionen und Notationen sind aus [Rei85] und [VDA11] übernommen.

Definition 2.2.1 (Petri-Netz [VDA11])
Ein Petri-Netz sei definiert als ein Triple N:

$$N = (S, T, F)$$

S und T sind endliche Mengen und es gilt:

$$S \cap T = \emptyset$$
$$F \subseteq (S \times T) \cup (T \times S)$$

S entspricht der Menge der Stellen, die Menge T beinhaltet alle Transitionen und F beschreibt die Flussrelation (entsprechend den Kanten im Graphen), sowohl von Stellen nach Transitionen als auch umgekehrt.

Multimengen [VDA11]

In der Prozessmodellierung reichen normale Mengen oft nicht aus. Zum Beispiel kann in einer Stelle eines Petri-Netzes nicht nur eine oder keine Marke sein, sondern auch mehrere. Daher ist es sinnvoll Multimengen einzuführen, bei denen ein Element auch eine Kardinalität besitzt oder mehrmals in einer Menge vorkommen kann. Die aufzählende Notation soll dazu mit „[“ und „]“ abgegrenzt werden. Mehrere gleiche Elemente können zu einem zusammengefasst werden, welches die Kardinalität in der Potenz trägt. Das folgende Beispiel beschreibt eine Multimenge M mit einem a, zwei b und drei c. Dabei sind die gegebenen Notationen äquivalent.

$$M = [a, b, b, c, c, c] = [a, b^2, c^2, c] = [a, b^2, c^3]$$

Anders kann eine Multimenge auch als ein Tupel $M = (D, \gamma)$ angesehen werden. Wobei D eine Menge und $\gamma : D \to \mathbb{N}_0$. M ist hierbei eine Multimenge über D und γ ordnet jedem Element eine Anzahl aus $\mathbb{N}_0$ zu. Nicht enthaltene Elemente werden dabei einfach auf die 0 abgebildet. Dieser Zusammenhang wird auch mit $M \in \mathbb{B}(D)$ annotiert.
Es folgt die Definition von Untermengen, welche mit $\leq$ annotiert werden: Für zwei Multimengen $A = (D, \gamma_A), B = (D, \gamma_B) \in \mathbb{B}(D)$ gilt

$$A \leq B \Leftrightarrow \forall x \in D : \ \gamma_A(x) \leq \gamma_B(x)$$

Die Summe zweier Multimengen $A = (D, \gamma_A), B = (D, \gamma_B) \in \mathbb{B}(D)$ sei definiert als Summe der Kardinalitäten:

$$A \biguplus B = (D, \gamma_{AB}), \quad \gamma_{AB}(x) = \gamma_a(x) + \gamma_b(x)$$

Definition 2.2.2 (Markiertes Petri-Netz [VDA11])
Ein markiertes Petri-Netz ist ein Tupel (N, M) bestehend aus einem Petri-Netz $N = (S, T, F)$ und einer Multimenge $M \in \mathbb{B}(S)$. Dabei stellt M die vorhandenen Marken dar und wird als Markierung bezeichnet.

Um auch mehrere Marken in einer Stelle modellieren zu können, ist M eine Multimenge. Die folgende Menge M_1 repräsentiert die Markierung aus Abbildung 2.3:

$$M_1 = [\textit{Aufgabe erzeugt}, \textit{Warteschlange}^3, \textit{Verbraucher bereit zum Empfangen}]$$

Oft ist für die Modellierung eines Prozesses ein markiertes Petri-Netz (N, M_0) gegeben. Dabei stellt M_0 die initiale Markierung des Modells dar.

Die folgende Definition beschreibt Vorgänger und Nachfolger in einem Petri-Netz:

Definition 2.2.3 (Vorgänger und Nachfolger [VDA11])
Sei ein Petri-Netz $N = (S, T, F)$ gegeben, dann sind folgende Mengen für $x \in S \cup T$ definiert:

$$\bullet x = \{y \,|(y, x) \in F\}$$
$$x\bullet = \{y \,|(x, y) \in F\}$$

$\bullet x$ beschreibt die Menge der Vorgänger und $x\bullet$ die Menge der Nachfolger des Elementes x. Falls $x \in S$, dann $\bullet x, x\bullet \subseteq T$ und analog dazu falls $x \in T$, dann $\bullet x, x\bullet \subseteq S$.

Definition 2.2.4 (Feuern einer Transition [VDA11])
Für eine Transition t eines markierten Petri-Netzes N, M, definieren wir die Relation $(N, M)[t\rangle$.

$$(N, M)[t\rangle \Leftrightarrow \bullet t \leq M$$

Falls diese Relation für eine gegebene Markierung erfüllt ist, dann wird die Transition als aktiv bezeichnet.

Wenn $(N, M)[t\rangle$ für beliebiges t gilt, dann sei folgende Feuerrelation definiert:

$$(N, M)[t\rangle(N, M') \Leftrightarrow M' = (M \backslash \bullet x) \biguplus x\bullet$$

$(N, M)[t\rangle(N, M')$ annotiert die Markierung, die entsteht, wenn die aktive Transition t gefeuert wird. Dabei wird von jeder Stelle aus $\bullet x$ eine Marke genommen und jeder Stelle aus $x\bullet$ einer Marke hinzugefügt. Zum Beispiel gilt für Abbildung 2.2 $(N, [\textit{links}])[a\rangle$, da in der abgebildeten Markierung a aktiv ist und $(N, [\textit{links}])[a\rangle(N, [\textit{rechts}])$, da beim Feuern von a die Marke aus *links* in *rechts* wandert.

Eine Feuersequenz ist die Liste von nacheinander ausführbaren Transitionen. Dies lässt sich mit der gegebenen Relation folgendermaßen definieren:

Definition 2.2.5 (Feuersequenz [VDA11])
Eine geordnete Multimenge $\sigma = \langle t_1, ..., t_n\rangle$ ist eine Feuersequenz von (N, M_0) falls die Markierungen $M_1, ..., M_n$ existieren für die gilt:

$$\forall i \in \{1, ..., n\} : (N, M_{i-1})[t_i\rangle \wedge (N, M_{i-1})[t_i\rangle(N, M_i)$$

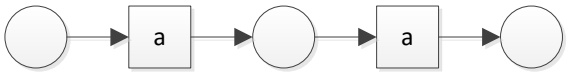


Abbildung 2.5.: Einfaches Beispiel, das die Notwendigkeit der Definition von beschrifteten Petri-Netzen zeigt.

Eine Markierung M gilt als erreichbar in einem markierten Petri-Netz N, M_0 wenn eine Feuersequenz von M_0 nach M existiert. Die Menge aller erreichbaren Markierungen wird mit $[N, M_0\rangle$ annotiert.

Definition 2.2.6 (Beschriftetes Petri-Netz [VDA11])
Ein beschriftetes Petri-Netz sei ein Tupel:

$$N = (S, T, F, A, b)$$

Dabei sei A eine Menge von Aktionsbeschriftungen und $b : T \to A$ eine Beschriftungsabbildung.

Der Sinn der beschrifteten Petri-Netze mag nicht einleuchten, da auch der Unterschied zu den bisherigen Netzen nicht groß ist. Sie dienen vor allem dazu, die Eins-zu-eins-Beziehung von Transition und der Beschriftung der Transition aufzulösen. Abbildung 2.5 zeigt ein Netz, welches mit der vorherigen Definition nicht korrekt wäre, da T eine Menge ist und nur einmal a enthalten kann. Um diese Einschränkung aufzuheben, werden Transitionen und ihre Beschriftungen getrennt. Somit sind die Transitionen aus Abbildung 2.5 unterschiedlich, werden aber auf dieselbe Beschriftung (Aktion) abgebildet. Somit ist es möglich die Ausführung von genau zweimal a hintereinander zu modellieren.

Da bei einem beschrifteten Petri-Netz jeder Transition explizit eine Beschriftung zugeordnet wird, die als die auszuführende Aktion angesehen werden kann, definieren wir noch die spezielle Beschriftung τ (Tau). Dieses Symbol steht explizit für keine Aktion und dient zur Modellierung eines Zustandsüberganges im Modell, der aber keine sichtbare Folge oder Aktion besitzt. Dies ermöglicht z. B. eine einfachere Darstellung einer bedingten Anweisung, bei der eine Aktion entweder ausgeführt wird oder nicht. Sie kann sozusagen übersprungen werden. Die τ-Beschriftung erweitert die Semantik der beschrifteten Petri-Netze und ermöglicht bessere und übersichtlichere Modellierungen.

Eigenschaften von Petri-Netzen

Es werden einige essenzielle Eigenschaften von markierten Petri-Netzen definiert [VDA11].

k-Beschränktheit Ein markiertes Petri-Netz (N, M_0) ist k-beschränkt, wenn keine mögliche erreichbare Markierung $M \in [N, M_0\rangle$ existiert, für die sich in einer Stelle mehr als k Marken befinden.

1-Beschränktheit Ein markiertes Netz ist sicher, wenn es 1-beschränkt ist, d. h. zu keinem späteren Zeitpunkt kann sich in einer Stelle mehr als eine Marke befinden.

Verklemmen Ein markiertes Petri-Netz ist frei von Verklemmen, wenn sich keine Markierung erreichen lässt, bei der keine Transition mehr feuern kann.

Lebendigkeit Eine Transition t eines markierten Petri-Netzes ist lebendig, wenn es eine mögliche Folge von ausführbaren Transitionen gibt welche t beinhaltet.

2.2.3. Workflow-Netze

Da Petri-Netze eine sehr weitreichende Semantik besitzen und damit sehr viele Konstrukte möglich sind, von denen einige nicht für die Betrachtung von Businessprozessen nötig sind, wird eine Untermenge der Petri-Netze eingeführt. Nach [VDA11] werden diese Workflow-Netze genannt. Diese sind Petri-Netze, welche eine dedizierte Startstelle und eine dedizierte Endstelle besitzen.

Definition 2.2.7 (Workflow-Netz [VDA11])
Sei $N = (S, T, F, A, b)$ ein beschriftetes Petri-Netz. N ist ein Workflow-Netz genau dann wenn:

1. *Es existiert eine Stelle $i \in S$ für die gilt $\bullet x = \emptyset$*
2. *Es existiert eine Stelle $o \in S$ für die gilt $x\bullet = \emptyset$*
3. *Für alle $x \in S \cup T$ existiert ein Pfad (in dem gerichteten Graphen) von i nach x.*
4. *Für alle $x \in S \cup T$ existiert ein Pfad (in dem gerichteten Graphen) von x nach o.*

Diese Petri-Netze besitzen jeweils eine Quelle (i) an der alle möglichen Abläufe starten und eine Senke (o) an der alle möglichen Aktionsabläufe enden.

Abbildung 2.6 zeigt ein Workflow-Netz, mit einer Startstelle, aus welcher nur Kanten entspringen, und einer Endstelle, die nur eingehenden Fluss besitzt und somit das Ende der Ausführung bedeutet. Um die schon gezeigten Petri-Netze einmal einzuordnen: Abbildung 2.2 und Abbildung 2.5 sind Workflow-Netze, während Abbildung 2.3 und Abbildung 2.4 keine sind.

Definition Korrektheit [VDA11]

Nach van der Aalst [VDA11] ist ein Workflow-Netz genau dann korrekt wenn folgende Bedingungen erfüllt sind:

Sicherheit $(N, [i])$ ist sicher, d. h. zu keinem Zeitpunkt beinhaltet eine Stelle mehr als eine Marke.

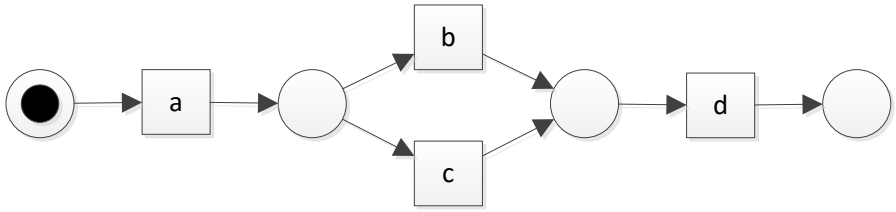


Abbildung 2.6.: Beispiel eines Workflow-Netzes.

Angemessene Beendigung Falls $o \in M$ für eine Markierung $M \in [N, [i]\rangle$, so ist $M = [o]$. Das bedeutet, falls sich eine Marke in der Senke befindet, ist die Ausführung sicher beendet, keine Transition ist mehr aktiv und es befindet sich auch keine Markierung in einer anderen Stelle.

Mögliche Beendigung $[o] \in [N, M\rangle$ für alle $M \in [N, [i]\rangle$.

Abwesenheit von toten Teilen Für $(N, [i])$ sind alle Transitionen lebendig.

2.3. Business Process Model and Notation (BPMN)

Eine andere weit verbreitete Notation zur Prozessmodellierung ist das Business Process Model and Notation (kurz BPMN). Die aktuelle Version 2.0 ist in [bpm11] spezifiziert. BPMN hat eine sehr reiche Semantik durch viele verschiedene Komponenten, welche einzelne allgemeine Eigenschaften und Ereignisse von Businessprozessen modellieren.

2.3.1. BPMN-Elemente

Abbildung 2.7 zeigt die für die Prozessmodellierung wichtigsten Elemente der BPMN. Die einzelnen Elemente werden durch gerichtete Kanten verbunden, welche den Kontrollfluss beschreiben. Dieser Fluss kann analog zu den Petri-Netzen als eine Bewegung von Marken durch den Graphen angesehen werden. Diese Marken fließen entlang der Kontrollflusskanten und werden an den Gateways entsprechend verändert. Gateways treten in zwei Varianten auf. Die erste hat die Aufgabe den Kontrollfluss aufzutrennen. Dies wird durch mehrere ausgehende Kanten annotiert. Entsprechend führt die zweite Variante den Kontrollfluss wieder zusammen. Das wird durch mehrere eingehende Kanten annotiert. Beide Varianten können auch in einem Gateway kombiniert werden.
Abbildung 2.8 zeigt ein BPMN, das die exklusive Ausführung von a oder b modelliert. Dabei spaltet das linke exklusive Gateway den Kontrollfluss auf genau einen Pfad auf. Das rechte exklusive Gateway führt die möglichen Ausführungspfade wieder zusammen.

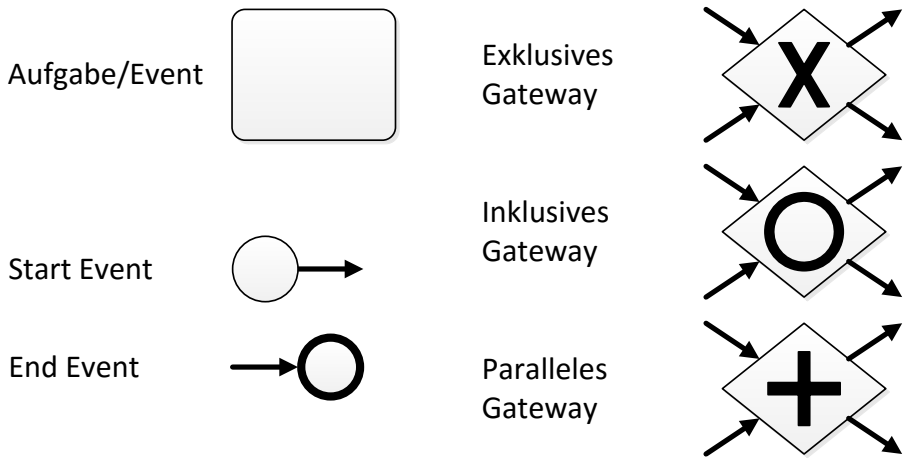


Abbildung 2.7.: Die Abbildung zeigt einen Ausschnitt der vorhandenen BPMN-Elemente.

Folgend werden BPMN-Elemente aus Abbildung 2.7 beschrieben:

Aufgabe (engl. Task) Repräsentiert eine atomare Aufgabe, die nicht weiter zu unterteilen und ein Teil des Prozesses ist.

Paralleles Gateway Das aufspaltende Gateway gibt den Kontrollfluss an alle ausgehende Kanten weiter. In der zusammenführenden Variante gibt das Gateway den Fluss erst weiter, wenn dieser aus allen eingehenden Kanten bereitsteht.

Exklusives Gateway In der aufteilenden Rolle gibt das Gateway den Kontrollfluss an genau eine ausgehende Kante weiter. Beim Zusammenführen wird jeder einzelne eingehenden Kontrollfluss propagiert.

Inklusives Gateway Das splittende Gateway propagiert den Fluss an mindestens eine ausgehende Kante und das zusammenführende Gateway wartet auf alle Flüsse, die auf Pfaden der eingehenden Kanten aktiv sind.

Startereignis Beginn des Kontrollflusses.

Endereignis Ende des Kontrollflusses.

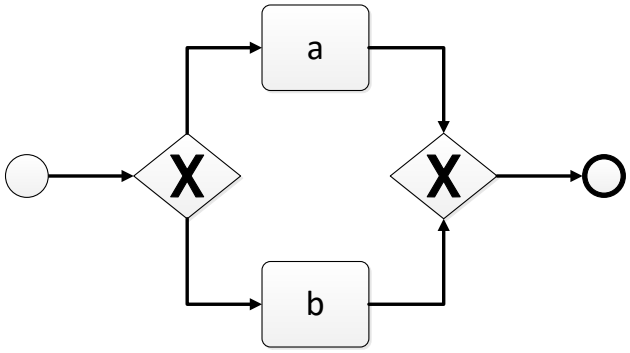


Abbildung 2.8.: Beispiel für ein simples BPMN, äquivalent zum Petri-Netz in Abbildung 2.2.

2.4. Information Retrieval

Für Software ist es leicht, jeden einzelnen Schritt einer Ausführung zu dokumentieren und auszugeben. Daher werden in Softwaresystemen viele Daten geloggt, um in einem Fehlerfall nachvollziehen zu können, was genau falsch gelaufen ist. Die Ausgaben der Programme werden meistens in Logdateien geschrieben. Eine Logdatei wird oft für einen bestimmten Zeitraum abgeschlossen (z. B. für jeden Tag eine) und jeder einzelne Eintrag in einem Log mit einem genauen Zeitstempel versehen. Zwar ist die Erstellung der Logs einfach, für das Extrahieren von Informationen, alleine schon das Finden einzelner Einträge für bestimmten Zeiten, ist das jedoch nicht praktikabel, denn die Dateien müssen dazu linear durchgesucht werden. Daher ist es sinnvoll die Einträge zu indexieren. Dazu werden vermehrt Suchmaschinen verwendet. Suchmaschinen passen gut zu Logdaten, da diese oft eine mangelnde Struktur besitzen, da die Nachricht in natürlicher Sprache verfasst ist.
Suchmaschinen bieten nun eine Menge an Operationen um die Logs zu durchsuchen. Zusätzlich lassen sich Metriken berechnen, die für das Data-Mining interessant sind.
Es existieren eine fertige Apache Lucene [luc15], welche das Implementieren eines solchen Systems unterstützen und sehr vereinfachen. Drauf baut die speziell für Programmdaten gedachte Sachmaschine Elasticsearch auf.

Elasticsearch, Kibana und Logstash

Elasticsearch ist eine Open-Source-Suchmaschine die für die schnelle Indizierung und Verarbeitung vom Applikationsdaten gedacht ist (siehe `https://www.elastic.co` und Abschnitt 2.4). Damit ist es unter anderem möglich in Echtzeit Performance-Indikatoren, wie die Anzahl an Aufrufen eines Service zu berechnen. Dazu werden meistens die durch die Applikationen erstellten Log-Daten an das System geschickt und indiziert. Zu diesem Zweck existiert unterstützend Logstash. Dies überwacht die

Logdateien der zu überprüfenden Anwendungen, strukturiert die Einträge mit Hilfe von regulären Ausdrücken und sendet die Daten an den Indexer des Elasticsearch. Kibana ist eine leichtgewichtige Webapp, welche einen schnellen und einfachen Zugriff auf die in Elasticsearch gespeicherten Daten bietet und Möglichkeiten zur Berechnung und Darstellung analytischer Werte (z. B. Anzahl bestimmter Einträge) stellt. Kibana bietet, nach dem Sinne von Information Retrival, auch eine Suche über den Logdaten an.

2.5. Process-Mining

Im idealisierten Konzept der Prozessmodellierung wird ein Modell von der Managementebene eines Unternehmens erstellt und eben dieses wird danach von betroffenen Personen dem Modell entsprechend umgesetzt und implementiert. In der reellen Welt ist dies aus verschiedenen Gründen leider oft nicht der Fall. Es kommt oft vor, dass kein Modell für einen Prozess existiert, da dieser einfach aus gegebenen Anforderungen erschaffen wurde und durch die praktische Ausführung optimiert wurde. Falls für einen Prozess ein Modell existiert, so ist dieses vielleicht nicht formal oder es existiert ein gutes formales Modell, die Ausführungen des Prozesses weichen aber von der Spezifikation ab. Diese Abweichungen können eine Optimierung der ausführenden Personen sein oder auch einfach eine willkürliche Handlung, die den Prozess verschlechtert.

Das Ziel von Process-Mining ist die Verbesserung oben genannter Probleme. Zum einen soll es die Erstellung von Modellen aus den Log-Daten der Instanziierung der Prozesse ermöglichen. Dieser Teil wird hauptsächlich in dieser Arbeit verwendet. Zum anderen bietet Process-Mining auch die Möglichkeit die Ausführung der Prozesse zu evaluieren, indem die Aufzeichnungen mit dem Modell durchgespielt werden und somit Konformität nachgewiesen werden kann. Außerdem ist es mithilfe der Daten nun auch möglich, das gegebene Modell zu verbessern. Zum Beispiel können Teile eines Modells entfernt werden, die in der Praxis nicht verwendet werden.

Process-Mining ermöglicht eine Vielzahl an verschiedene Ansichten auf die Daten der ausgeführten Prozesse. Zum Beispiel können die Interaktionen zwischen den beteiligten Personen betrachtet werden [WF94]. Eine andere Möglichkeit ist eine zeitliche Analyse, bei der Ausführungsdauer und Wartezeiten analysiert und optimiert werden können. Auch kann der Verbrauch von Ressourcen betrachtet werden. Die höchste Komplexität bietet die Kontrollflussansicht, bei der die Abhängigkeiten von Aufgaben und deren relative Ausführungsreihenfolge betrachtet werden. Das Erstellen von Ablaufmodellen wird im Process-Discovery genannt und stellt die Technik aus dem Process-Mining dar die in dieser Arbeit verwendet wird.

Cappuccino	Latte	Espresso	Americano	Ristretto	Tee	Muffin	Bagel
1	0	0	0	0	0	1	0
0	2	0	0	0	0	1	1
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	1	2	0
0	0	0	1	1	0	0	0
...	...	...	...	...	...	...	...

Tabelle 2.1.: Datensatz der Bestellungen eines Cafés aus [VDA11].

Process-Mining ist in Relation zu gegeben anderen Disziplinen nach van der Aalst [VDA11] zwischen Prozessmodellierung und Data-Mining anzuordnen.

2.5.1. Data-Mining

Data-Mining ist eine Wissenschaft, welche die Möglichkeiten untersucht, aus großen Mengen von Daten verwertbare Informationen zu extrahieren. Das Ziel dabei sind häufig aggregierte Werte, erkannte Muster oder Modelle, welche den Daten zugrunde liegen könnten.

Tabelle 2.1 zeigt einen Ausschnitt eines Datensatzes [VDA11], der die Bestellungen eines Cafés darstellt. Diese Daten wurden zur Dokumentation der Verkäufe erhoben. Mit Hilfe von Data-Mining lassen sich diese nutzen, um das Verhalten der Nutzer des Cafés zu analysieren und Abläufe zu verbessern. Zum Beispiel können Beziehungen zwischen Produkten erkannt werden, welche Produkte oft und welche niemals zusammen gekauft werden. Mithilfe solcher Zusammenhänge lässt sich das Verhalten der Kunden besser vorhersagen. Eine einfache Analyse könnte sein, dass Kunden, die Tee trinken, oft einen Muffin bestellen. Data-Mining versucht Muster in den Daten oder, im Fall des Process-Mining, Modelle zu finden, welche die Daten erzeugt haben könnten, um die Realität besser verstehen zu können.

2.5.2. Logs, Traces und Events

Bei der Dokumentation von Prozessen wird oft für jedes Ereignis eine Menge an Eigenschaften gespeichert. Oft sind es mindesten ein Zeitstempel, das Ereignis und eine ID welche die Zugehörigkeit zu einem Vorgang beschreibt. (Dies alles ist unter der Annahme, dass die Einträge zu genau einem Prozess gehören. Falls dies nicht der Fall ist, wird auch eine Zuordnung zu einen definierten Prozess benötigt.) Weiterhin können dies Eigenschaften wie eine eindeutige ID des Eintrages, die Lebenszyklus-Veränderung des Ereignisses für den Eintrag (z.B. Start, Ende), Ressourcen oder beteiligte Personen sein. Das Vorhandensein von

Informationen in den Logs ist sehr abhängig von dem System im Einzelnen. Mithilfe aller dieser Eigenschaften und Techniken des Data-Mining können viele Informationen extrahiert werden. Diese Arbeit betrachtet aber nur die Kontrollflussrelation der Einträge. Dafür lassen sich die Einträge auf ihre zeitliche Relation pro Ablauf reduzieren. Es werden die folgenden Begriffe verwendet [VDA11]:

Event Ereignis welches eine atomare Aktion beschreibt. In den bisherigen Beispielen wäre die Ausführung von a oder einfach nur a ein Event.

Trace Eine Liste von **Events** für welche die Kontrollflussrelation definiert ist. D.h. ihre Ausführungsreihenfolge ist bekannt. Zum Beispiel eine Liste mit den Events a und b, wobei a vor b ausgeführt wird. Ein Trace wird mit $\langle$ und $\rangle$ annotiert und die Reihenfolge beschreibt die Kontrollflussrelation. Für das Beispiel wäre das $\langle a, b \rangle$.

Log Eine Multimenge von **Traces**, die zum selben Prozess gehören. Formal wird ein Log über einer Menge $\mathcal{A}$ definiert, welche die möglichen Events beschreibt. In der Praxis wird $\mathcal{A}$ als die Menge aller im Log dokumentierter Events angenommen. Ein Log, welcher aus der Ausführung von Abbildung 2.6 entstanden sein könnte, wäre zum Beispiel:

$$L = [\langle a, b, d \rangle^7, \langle a, c, d \rangle^4]$$

2.5.3. Extensible Event Stream (XES)

Extensible Event Stream (XES) ist XML basiertes Datenformat und standardisiertes Dateiformat zur Beschreibung von Logs [GV14]. Dabei werden die oben beschriebenen Kontrollflussrelationen beschrieben und zusätzlich können noch weitere Eigenschaften auf Ebene der Events, der Traces oder des Logs annotiert werden. Somit sind grundlegende Eigenschaften sichergestellt und gleichzeitig können beliebige zusätzliche Informationen hinzugefügt werden.

Die Struktur einer XES-Datei entspricht der Hierarchiestruktur eines Logs im Allgemeinen (siehe Unterabschnitt 2.5.2). Das XML-Wurzelelement repräsentiert den Log. Es existiert daher genau ein Log pro Datei. Der Log enthält eine beliebige Anzahl an Traces und dieser wiederum eine beliebige Anzahl an Events. Dabei können allen Elementen noch zusätzlich Attribute angehängt werden. Diese definieren bestimmte Eigenschaften der Elemente. Einige sind im Standard vordefiniert, es können aber beliebige weitere definiert werden um den Log mit Informationen anzureichern, ohne die Kompatibilität einzuschränken. Einige Attribute, wie etwa den Namen eines Events, lassen sich nicht weglassen, ohne die Nutzbarkeit der Daten einzuschränken. In diesem Beispiel des Namens würde ein Bezeichner fehlen, wodurch viele Techniken des Process-Mining wie die Process-Discovery (siehe Abschnitt 2.6) nicht möglich sind.

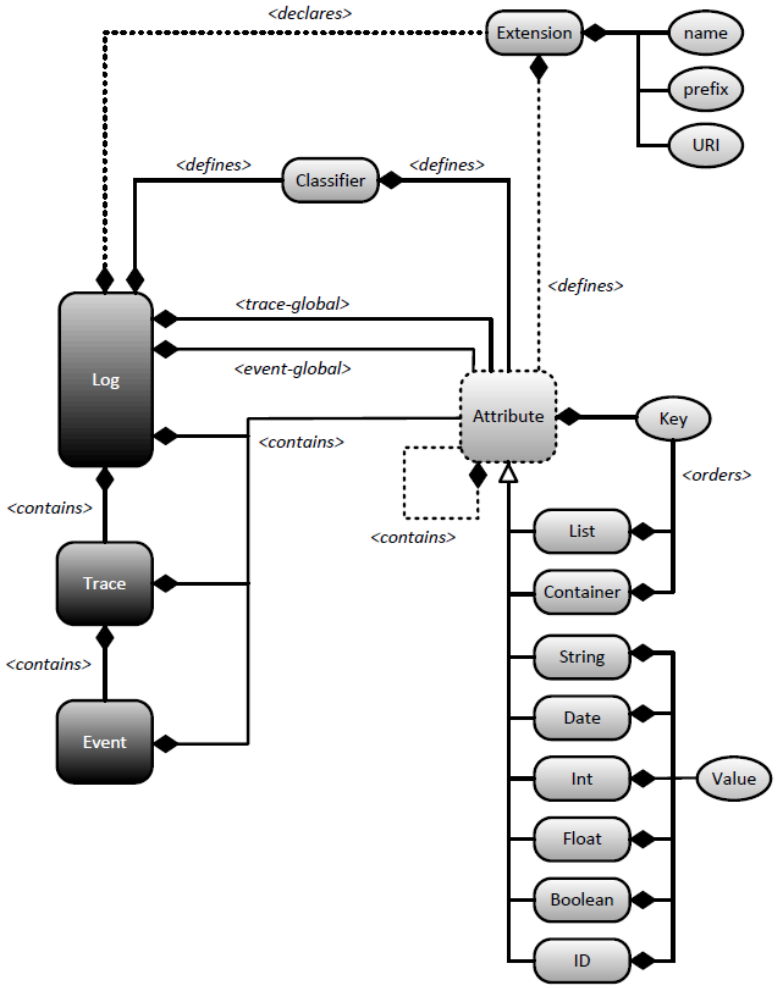


Abbildung 2.9.: XES-Schema in UML-2.0-Notation [GV14].

2.6. Process-Discovery

Bei der Process-Discovery wird der Fall betrachtet, dass für ein System oder einen Prozess kein Kontrollflussmodel gegeben ist. Das Ziel ist es aus aufgenommenen Daten, einem Log, ein Modell zu extrahieren, das möglichst gut die gesehenen Abläufe erzeugt haben könnte. Da im Allgemeinen nicht einmal ein eindeutiges optimales Modell existiert (abgesehen von einigen einfachen Beispielen), ist das Finden eines geeigneten Algorithmus nicht trivial und es existiert eine Vielzahl von Ansätzen.

Van der Aalst definiert die **allgemeine** Herausforderung der Process-Discovery, einen Algorithmus zu finden, der für einen Log ein "repräsentatives“ Modell findet [VDA11]. Ein solcher Algorithmus bekommt einen Log als Eingabe und liefert ein Prozessmodell als Ausgabe.

Formal ist es erst einmal unerheblich, in welcher Notation sich das Modell befindet, da die Algorithmen leicht für andere Notationen angepasst werden können und weil es oft möglich ist die unterschiedlichen Notationen ineinander zu überführen. Van der Aalst [VDA11] beschreibt das **spezielle** Problem der Process-Discovery einen Algorithmus zu finden, der als Ausgabe ein korrektes Workflow-Netz liefert und sich mit diesem Netz jeder einzelne der gegebenen Traces erzeugen lässt. Zwar sollten alle im Log vorkommenden Traces mit dem Modell ausführbar sein, neue, nicht im Log vorhandene, Abläufe dürfen aber durch das Modell erzeugt werden. Dies ist für ein gutes Modell sogar meistens nötig. Zum Beispiel, im Falle eines Zyklus, sollte sich dieser auch im Modell widerspiegeln. Aufgrund der Endlichkeit des Logs können jedoch nicht möglichen Traces vorhanden sein. Dadurch kann das Modell zusätzliche Sequenzen erzeugen. Außerdem ist es im Allgemeinen nötig zu generalisieren, da sonst das Modell unnötig komplex und nicht mehr verwendbar wird. Trotzdem ist es auch unbrauchbar zu viel neue Abläufe zu erlauben, da das Modell sonst seine Aussagekraft verliert. Das erzeugte Netz sollte eine gute Abwägung verschiedener Qualitätsindikatoren erfüllen.

2.6.1. Qualitätsindikatoren

Um zu bewerten wie geeignet ein Netz für gegebene Traces ist, werden vier Qualitätskriterien eingeführt [VDA11]:

Eignung (eng. fitness) Kriterium wie viele der gesehenen Verhalten sich mit dem Modell nachspielen lassen. Ein Petri-Netz mit einer hohen Eignung kann die meisten der geloggten Sequenzen durch Feuersequenzen nachspielen.

Genauigkeit (eng. precision) Je weniger Verhalten das Modell zulässt, das sich nicht in den Traces widerspiegelt, desto genauer ist das Modell. Ein vollkommen genaues Modell würde nur Verhalten zulassen, das im Log sichtbar ist.

Generalisierung (eng. generalization) Ein Mining-Algorithmus sollte aus Sequenzen ein allgemeines Verhalten ableiten. Leicht verschieden Traces führen bei einer guten Generalisierung nur zu kleinen Verzweigungen. Im Gegensatz dazu könnte ein schlecht generalisierendes Modell komplette Traces als Möglichkeiten modellieren (siehe Abbildung 2.12).

Einfachheit (eng. simplicity) Ein gutes Modell sollte so einfach wie möglich sein. Die Einfachheit eines Modelles kann gut über die Anzahl der verwendeten Elemente geschätzt werden. Ein einfacheres Modell lässt sich besser durch einen Nutzer verstehen.

Da diese Kriterien sich zum Teil gegenseitig widersprechen, ist es nicht einfach, für einen Algorithmus alle Kriterien zu optimieren. Oft ist es sehr einfach einen Algorithmus zu finden, der ein Kriterium voll erfüllt, aber dafür in anderen schlecht abschneidet und daher meistens eher unbrauchbare Modelle liefert. Zum Beispiel wäre ein Algorithmus, der die Eignung

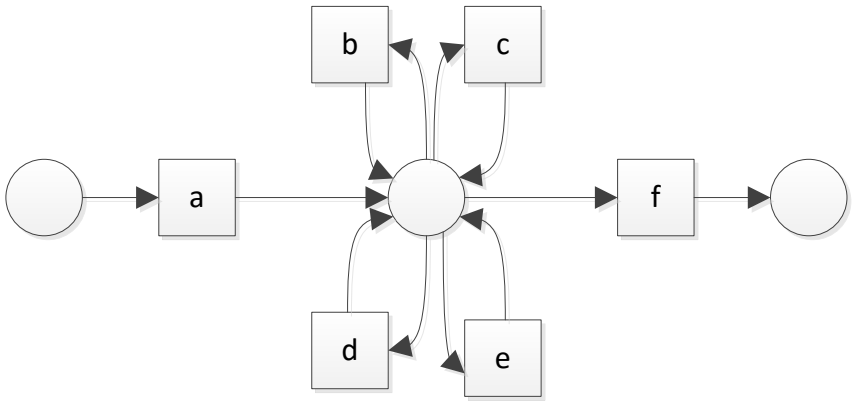


Abbildung 2.10.: Blumen Petri-Netz, das eine schlechte Genauigkeit gegeben über L_1 aufweist.

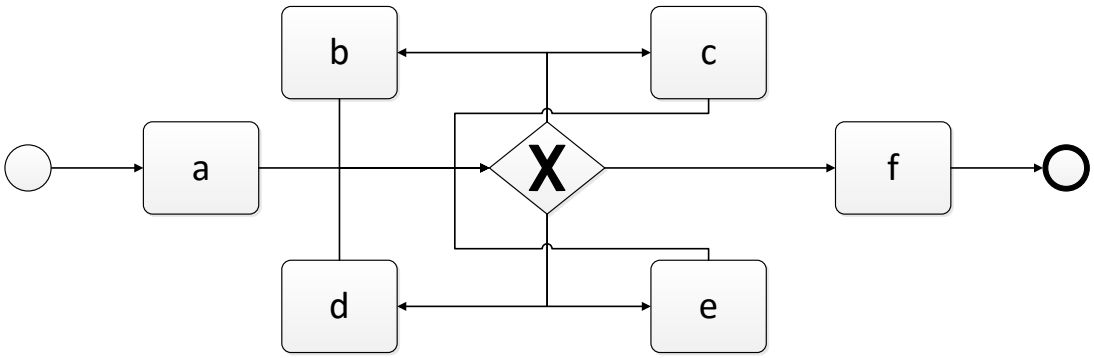


Abbildung 2.11.: Blumen BPMN mit schlechter Genauigkeit analog zu Abbildung 2.10.

für ein Petri-Netz optimieren würde, einfach die unterschiedlichen Sequenzen als komplett getrennten lineare Pfade des Netzes zu erstellen. Dadurch wird immer eine maximale Eignung und Genauigkeit erreicht, das Netz ist jedoch meisten nicht sehr generalisiert oder einfach.

$$L_2 = [\langle a, b, d, e\rangle, \langle a, c, e\rangle, \langle a, b, c, d, e\rangle, \langle a, b, c, e\rangle, \langle a, d, e\rangle, \langle a, c, d, e\rangle]$$

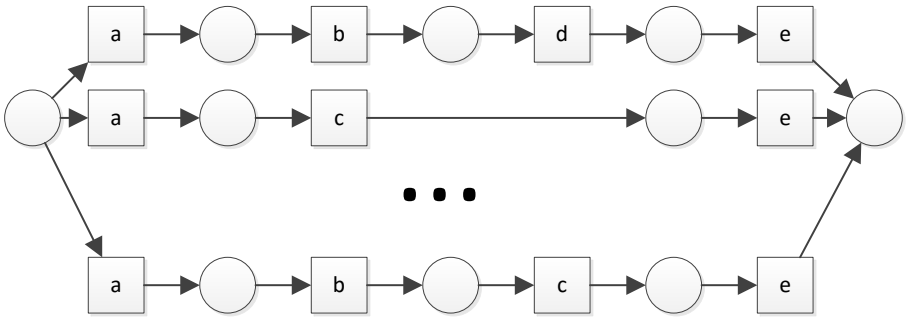


Abbildung 2.12.: Petri-Netz mit allen Pfaden, das eine perfekte Genauigkeit für L_2 hat.

2.6.2. α-Algorithmus

Für den α-Algorithmus müssen zuerst Log basierte Event-Ordnungsrelationen definiert werden nach [AWM04] und [VDA11].

Für einen Log L werden folgende Relationen definiert:

- $a >_L b \Leftrightarrow \exists T = \langle t_1, ..., t_n \rangle \in L, \exists i \in \{1, ..., n-1\} : t_i = a \wedge t_{i+1} = b$.
  $a >_L b$ gilt genau dann, wenn ein Trace in L existiert in dem b direkt auf a folgt.
- $a \rightarrow_L b \Leftrightarrow a >_L b \ \ \wedge \ \ b \not>_L a$
- $a \#_L b \Leftrightarrow a \not>_L b \ \ \wedge \ \ b \not>_L a$
- $a \parallel_L b \Leftrightarrow a >_L b \ \ \wedge \ \ b >_L a$

Zum Beispiel sei ein Log L_1 über $\mathcal{A} = \{a, b, c, d\}$ gegeben.

$$L_1 = [\langle a, c, b, d \rangle^5, \langle a, b, c, d \rangle, \langle a, b, c \rangle^2]$$

Daraus ergeben sich die folgenden Relationen.

$$\begin{aligned} >_{L_1} = &\quad \{(a,b),(a,c),(b,c),(b,d),(c,b),(c,d)\} \\ \rightarrow_{L_1} = &\quad \{(a,b),(a,c),(b,d),(c,d)\} \\ \#_{L_1} = &\quad \{(a,a),(a,d),(b,b),(c,c),(d,a),(d,d)\} \\ \parallel_{L_1} = &\quad \{(b,c),(c,b)\} \end{aligned}$$

Für alle Paare $x, y \in \mathcal{A}$ gilt genau eine der folgenden Relationen $x \rightarrow_{L_1} y, y \rightarrow_{L_1} x, x \#_{L_1} y, x \parallel_{L_1} y$. Daher kann dieser Zusammenhang in einer Matrix dargestellt werden (siehe Tabelle 2.2).

	a	b	c	d
a	$\#$	$\rightarrow$	$\rightarrow$	$\#$
b	$\leftarrow$	$\#$	$\|\|$	$\rightarrow$
c	$\leftarrow$	$\|\|$	$\#$	$\rightarrow$
d	$\#$	$\leftarrow$	$\leftarrow$	$\#$

Tabelle 2.2.: Relationen für L_1.

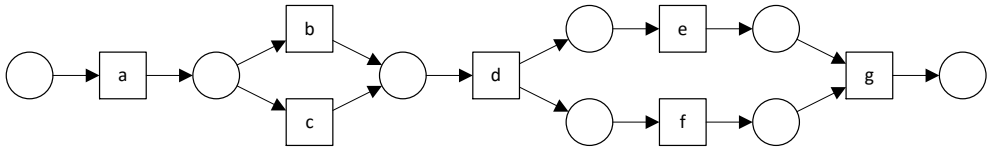


Abbildung 2.13.: Petri-Netz für L_2.

Mithilfe dieser Relationen lassen sich nun einzelne Trennungs- und Zusammenführungskonstrukte herauslesen. Dies wird an dem Beispiel L_2 verdeutlicht.

$$L_2 = [\langle a, b, d, e, f, g\rangle^{10}, \langle a, c, d, f, e, g\rangle^{7}, \langle a, b, d, f, e, g\rangle^{16}, \langle a, c, d, e, f, g\rangle^{5}]$$

Das zugrunde liegende Petri-Netz ist in Abbildung 2.13 zu sehen. Von den gegeben Traces diese Netz zu rekonstruieren, stellt für einen Menschen eine bewältigbare Aufgabe dar, es ist jetzt zu betrachten, wie sich dies mithilfe der definierten Relationen verallgemeinern lässt. Dazu steht die Relationsmatrix in Tabelle 2.3.

Für die Events b und c wird der Workflow exklusiv gespalten, d.h. es einer bestimmten Stelle genau eine der beiden Aktionen durchgeführt. Das Teilen kann durch die Relationen

	a	b	c	d	e	f	g
a	$\#$	$\rightarrow$	$\rightarrow$	$\#$	$\#$	$\#$	$\#$
b	$\leftarrow$	$\#$	$\#$	$\rightarrow$	$\#$	$\#$	$\#$
c	$\leftarrow$	$\#$	$\#$	$\rightarrow$	$\#$	$\#$	$\#$
d	$\#$	$\leftarrow$	$\leftarrow$	$\#$	$\rightarrow$	$\rightarrow$	$\#$
e	$\#$	$\#$	$\#$	$\leftarrow$	$\#$	$\|\|$	$\rightarrow$
f	$\#$	$\#$	$\#$	$\leftarrow$	$\|\|$	$\#$	$\rightarrow$
g	$\#$	$\#$	$\#$	$\#$	$\leftarrow$	$\leftarrow$	$\#$

Tabelle 2.3.: Relationen für L_2.

Algorithmus 2.1 α-Algorithmus aus [AWM04]

Eingabe: Ein Log L als eine Multimenge von Traces.

$$
\begin{aligned}
T_L &= \{t \mid \exists \sigma \in L : t \in \sigma\} \\
T_I &= \{t \mid \exists \sigma = \langle t_1, ..., t_n \rangle \in L : t = t_1\} \\
T_O &= \{t \mid \exists \sigma = \langle t_1, ..., t_n \rangle \in L : t = t_n\} \\
X_L &= \{(A, B) \mid A \subseteq T_L \wedge A \neq \emptyset \wedge B \subseteq T_L \wedge B \neq \emptyset \wedge \forall a \in A \, \forall b \in B : a \rightarrow_L b \\
&\quad \wedge \forall a_1, a_2 \in A : a_1 \# a_2 \wedge \forall b_1, b_2 \in B : b_1 \# b_2\} \\
Y_L &= \{(A, B) \in X_L \mid \forall (A', B') \in X_L : A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B')\} \\
P_L &= \{p(A, B) \mid (A, B) \in Y_L\} \cup \{i_L, o_L\} \\
F_L &= \{(a, p(A, B)) \mid (A, B) \in Y_L \wedge a \in A\} \\
&\quad \cup \{(p(a, b), b) \mid (A, B) \in Y_L \wedge b \in B\} \\
&\quad \cup \{(i_L, t) \mid t \in T_I\} \\
&\quad \cup \{(t, o_L) \mid t \in T_O\} \\
\alpha(L) &= (P_L, T_L, F_L)
\end{aligned}
$$

Ausgabe: Ein Workflow-Netz $\alpha(L)$.

$a \rightarrow b$, $a \rightarrow c$ und $b \# c$ beschrieben werden. Da auf a b oder c folgen kann und diese nicht in direkter Nachbarschaft vorkommen, handelt es sich um exklusive Verzweigung. So ähnlich ist es auch bei der Zusammenführung, für welche die Relationen $b \rightarrow d$, $c \rightarrow d$ und $b \# d$ gelten. Die Relationen dienen als Grundlage um die entsprechenden Petri-Netz Konstrukte zu bestimmen. Für die parallel Ausführung sind das analog $d \rightarrow e$, $d \rightarrow f$ und $e \parallel f$ für die Trennung und $e \rightarrow g$, $f \rightarrow g$ und $e \parallel f$ für die Zusammenführung. Diese Erkenntnisse sind die Grundlage des α-Algorithmus.

Algorithmus 2.1 beschreibt den α Algorithmus. Dieser beinhaltet die oben besprochene Idee im 4. Schritt bei der Erstellung von X_L. Um einzeln die Schritte durchzugehen: T_L beschreibt die Menge aller Transitionen, die durch die Menge aller in dem Log vorkommenden Events definiert wird. T_I und T_O beschreiben die jeweiligen Transitionen, deren entsprechende Events zu Beginn / Ende eines Trace aufgetreten sind. Die Bedingungen für die Paare von Mengen in X_T beinhaltet die Idee der Bedingungen für eine exklusives Zusammenführen und Aufteilen. Die parallele Relation wird nur indirekt benutzt, indem die Elemente aus den einzelnen Mengen mit $\#$ in Relation stehen müssen, d.h. dass diese Ereignisse nie in direkter Nachbarschaft aufgetreten sind. Die Menge Y_L ist eine Optimierung von X_L, welche nur die maximalen Paare beinhaltet. Alle (A, B) für die ein anderes (A', B') in X_L existiert, für das entweder A eine echte Teilmenge von A' ist oder B eine echte Teilmenge von B', werden entfernt. Für jedes Paar aus Y_L wird nun eine Stelle erstellt. Diese Menge an Stellen plus jeweils eine für Start und Ende ergeben P_L. F_L beschreibt die Mengen an Kontrollflussrela-

tionen, welche für $(A, B) \in Y_L$ für die entsprechenden Stellen durch eingehenden Fluss von allen Transitionen aus A und als ausgehenden Fluss zu alle Transitionen aus B beschrieben werden. Start und Ende werden zusätzlich mit den am Anfang bestimmten ersten und letzten Transitionen aus T_I und T_O verbunden. Das Triple (P_L, T_L, F_L) beschreibt das durch den Algorithmus erzeugte Netz.

Um die Arbeitsweise des Algorithmus zu zeigen, wird er auf L_2 angewendet.

$$
\begin{aligned}
T_L &= \{a, b, c, d, e, f, g\} \\
T_I &= \{a\} \\
T_O &= \{g\} \\
X_L &= \{(\{a\}, \{b\}), (\{a\}, \{c\}), (\{a\}, \{b, c\}), (\{b\}, \{d\}), (\{c\}, \{d\}), (\{b, c\}, \{d\}), \\
&\quad (\{d\}, \{e\}), (\{d\}, \{f\}), (\{e\}, \{g\}), (\{f\}, \{g\})\} \\
Y_L &= \{(\{a\}, \{b, c\}), (\{b, c\}, \{d\}), (\{d\}, \{e\}), (\{d\}, \{f\}), (\{e\}, \{g\}), (\{f\}, \{g\})\} \\
P_L &= \{i_L, p(\{a\}, \{b, c\}), p(\{b, c\}, \{d\}), p(\{d\}, \{e\}), p(\{d\}, \{f\}), \\
&\quad p(\{e\}, \{g\}), p(\{f\}, \{g\}), o_L\} \\
F_L &= \{(i_L, a), (a, p(\{a\}, \{b, c\})), (p(\{a\}, \{b, c\}), b), (p(\{a\}, \{b, c\}), c), \\
&\quad (b, p(\{b, c\}, \{d\})), (b, p(\{b, c\}, \{d\})), (p(\{b, c\}, \{d\}), d), (d, p(\{d\}, \{e\})), \\
&\quad (d, p(\{d\}, \{f\})), (p(\{d\}, \{e\}), e), (p(\{d\}, \{f\}), f), (e, p(\{e\}, \{g\})), \\
&\quad (f, p(\{f\}, \{g\})), (p(\{e\}, \{g\}), g), (p(\{f\}, \{g\}), g), (f, o_L)\}
\end{aligned}
$$

Dies entspricht, bis auf die fehlenden Bezeichnungen der Stellen, dem Netz aus Abbildung 2.13. Die Bezeichnungen der Stellen können jedoch aus dem Kontext inferiert werden und würden nur der Übersichtlichkeit schaden. Daher wird die Bezeichnung der Stellen im Allgemeinen auch weggelassen.

2.6.3. Grenzen des α-Algorithmus

Es lässt sich für einen Algorithmus nicht direkt bestimmen, ob er das korrekte Model für einen Log erzeugt. Auf Grund der verschiedenen Qualitätskriterien gibt es eine Menge an Modellen, welche den zugrunde liegenden Prozess gut beschreiben. Daher kann ein solches Verhalten nur zum Teil untersucht werden, in dem ein Modell erstellt wird, aus diesem Traces extrahiert werden und schlussendlich betrachtet wird, ob der Algorithmus ein äquivalentes Modell erzeugt. Eine ausführliche Betrachtung dieses Ansatzes findet sich in [VDA11, 140-147]. Hier werden einige Beispiele betrachtet, bei denen der α-Algorithmus das zugrunde liegende Modell nicht zurückgewinnen kann.

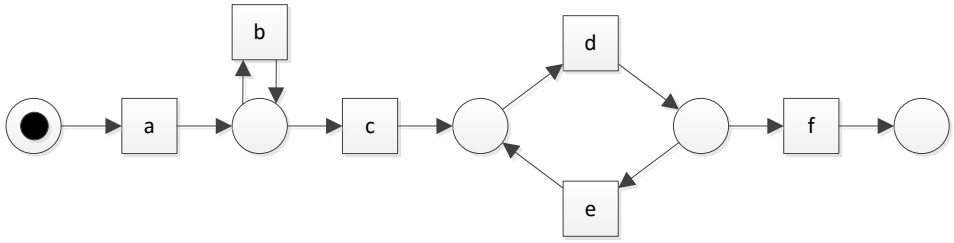


Abbildung 2.14.: Workflow-Netz mit Schleifen der Größe 1 und 2.

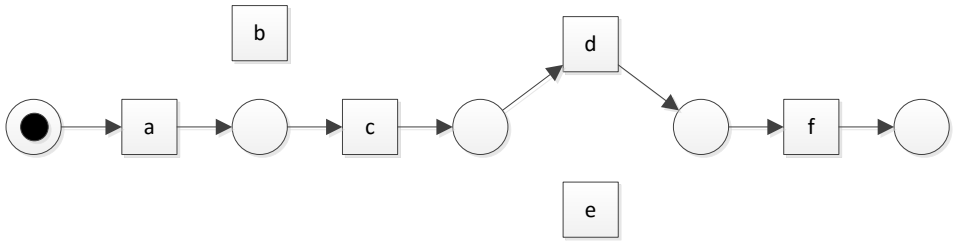


Abbildung 2.15.: Workflow-Netz, das der Alpha Algorithmus aus L_3 erstellt.

Probleme mit Schleifen der Länge 1 und 2

Der α-Algorithmus hat Probleme mit Schleifen der Länge 1 und 2, d.h. er erkennt diese nicht und erstellt ein inkorrektes Workflow-Netz. Dieses Verhalten lässt sich am Besten an einem Beispiel zeigen. Gegeben sei das Workflow-Netz aus Abbildung 2.14 und ein möglicher Log L_3 mit einigen Traces die mit diesem Netz erzeugt werden können.

$$L_3 = [\langle a, c, d, f\rangle, \langle a, b, c, d, f\rangle, \langle a, c, d, e, d, f\rangle, \langle a, b, b, c, d, e, d, f\rangle, \langle a, b, c, d, e, d, e, d, f\rangle]$$

Abbildung 2.15 zeigt das Modell, welches durch den Algorithmus erstellt wird. Es weißt fehlende Kanten an den Transitionen b und e auf. Formal bedeutet dies, dass diese Transitionen zu jedem Zeitpunkt ausgeführt werden können, da sie keine Abhängigkeiten besitzen. Dies widerspricht den Forderungen an ein Workflow-Netz und offensichtlich auch dem Log L_3. Bei der Transition b handelt es sich um das Problem, das bei Schleifen der Länge 1 auftritt. Da im Log b auf b folgt, gilt $b \parallel_{L_3} b$ und somit kann b im Schritt 4 des Algorithmus niemals in einer der Mengen A oder B vorkommen. Somit erhält b keine Kanten. Ähnlich verhält es sich auch mit e, hier gilt $d \parallel e$ und somit kann d nicht vor oder nach e kommen und die nötigen Kanten fallen weg. Eine Erweiterung des α-Algorithmus der so genannte $\alpha+$-Algorithmus wird von de Medeiros et al. in [MAW03] beschrieben. Durch Einführung von Vor- und Nachbearbeitungsphasen werden diese Art von Schleifen erkannt und bearbeitet.

Probleme mit nicht lokalen Abhängigkeiten

Der α-Algorithmus ist sehr lokal eingeschränkt. Alle Erkenntnisse werden immer aus der direkten Nachbarschaft von Aktionen im Log erhoben. Daher ist es dem Algorithmus nicht möglich nicht lokale Abhängigkeiten zu erkennen. Oft ist dies in der Realität jedoch relevant, da ein Prozess je nachdem durch was er eingeleitet wurde auch anders endet oder weiter geht. Dies wird anhand des Beispiel-Logs L_4 erklärt.

$$L_4 = [\langle a, b, c\rangle, \langle d, b, e\rangle]$$

Der α-Algorithmus arbeitet nur auf den Relationen zu den direkten Nachbarn. Daher erkennt er nicht, ob vor dem b ein a oder d war. Er kann auch nicht dem b einem bestimmten Kontext mitgeben (mit dem sich ein b von einem anderen b unterscheiden lies), mit dessen Hilfe dies bestimmt werden könnte. Daher erzeugt der Algorithmus ein Modell, dass auch Abfolgen wie $\langle a, b, e\rangle$ zulässt.

Auch hier gibt es Ansätze den Algorithmus zu verbessern, aber diese Probleme sollen vor allem aufzeigen, dass der Algorithmus nur eine gute Grundlage für die Process-Discovery bietet. Daher gibt es schon eine Menge an weiteren Ansätzen, die das Process-Discovery Problem zu lösen versuchen. Diese Arbeit betrachtet einen weiteren Ansatz, welcher mit Hilfe von evolutionären Algorithmen das Process-Discovery-Problem löst.

2.6.4. Evolutionärer Ansatz

Evolutionäre oder genetische Algorithmen stammen aus dem Teilgebiet der künstlichen Intelligenz und versuchen Probleme zu lösen, in dem sie eine aus der Biologie stammend Idee auf die Informatik übertragen. Der Algorithmus funktioniert nach dem Prinzip des „Überleben der Stärksten“. Als Erstes werden Modelle zufällig erstellt. Dann werden die Modelle wiederholt kombiniert, mutiert und ungeeignete Instanzen entfernt. Dieser ganze Prozess konvergiert dabei zu einem repräsentativen Prozessmodell.
Das Prinzip der evolutionären Algorithmen hat sich schon in vielen Problemen, z. B. beim Investment Banking, als nicht nur ein sehr guter Ansatz bewiesen, sondern auch als zum Teil anderen Ansätzen bei weitem überlegen [ES15].

Algorithmus 2.2 zeigt den Ablauf eines evolutionären Algorithmus. Eine ausführlicher Einführung ist in [ES15] nachzulesen. Für eine spezielle Implementierung zur Lösung eins Problems werden die einzelnen Schritte nur gewählt oder speziell implementiert. Im Allgemeinen laufen diese jedoch ähnlich ab und können somit schnell auf andere Probleme angewendet werden:

Initialisierung Die Initialisierung ist das initiale Erstellen der Population. Dabei werden meistens zufällige Lösungsinstanzen verwendet. Im Fall der Process-Discovery ist es sinnvoll initiale Modelle zu verwenden, welche zumindest alle vorkommenden

Algorithmus 2.2 Das generische Schema eines evolutionären Algorithmus aus [ES15]

INITIALISIERE Population mit einer Menge aus zufälligen Lösungen.
BEWERTE jeden Kandidaten
WIEDERHOLE BIS (*ABBRUCHBEDINGUNG* wahr)
- *WÄHLE* Eltern
- *KREUZE* Paare der Eltern
- *MUTIERE* die entstandenen Nachfahren
- *BEWERTE* die neuen Kandidaten
- *WÄHLE* die Lösungen für die nächste Generation

OD

Events beinhalten, wodurch der Arbeitsaufwand verringert wird, da schneller geeignete Modelle entstehen.

Bewertung Um nur die Stärksten überleben zu lassen und Schwächere auszusortieren, muss definiert sein, welche Lösungsinstanzen besser und welche schlechter sind. Daher ist es nötig eine Abbildung zu definieren, welche die Eignung der Lösungen bewertet.

Auswählen der Eltern Es werden nur ein Teil der Population als Eltern für die nächste Generation gewählt. Meistens werden nicht einfach nur die Besten genommen, sondern die Eltern probabilistisch gewählt, wobei die Wahrscheinlichkeit trotzdem proportional zur Eignung der Instanzen ist. Somit werden schlechte Instanzen mit einer kleinen Wahrscheinlichkeit, größer als Null, trotzdem als Eltern gewählt.

Kreuzen Hierbei werden zwei oder mehr Eltern genommen um diese zu einer neuen Instanz zu kombinieren. Hierbei werden Teile aus beiden Eltern übernommen. Diese Instanz kann geeigneter oder ungeeigneter als die Vorfahren sein.

Mutation Die entstanden Nachfahren werden noch mit einer gewissen Wahrscheinlichkeit zufällig mutiert. D. h. es ist zufällig, ob die Mutation durchgeführt wird und zufällig, was die Mutation ändert.

Wählen der nächsten Generation In den meisten Fällen ist die Größe der Population beschränkt und die neue wird aus den Besten der alten Generation und den neuen Kandidaten gewählt.

Abbruchbedingung Für die Abbruchbedingung können verschiedene Kriterien gewählt werden. Die optimale Bedingung wäre abzubrechen, wenn eine Lösung gefunden ist, die eine **Bewertung** von $100\ \%$ aufweist. Da dies in vielen Fällen gar nicht erreicht werden kann oder für eine Lösung nicht bestimmt werden kann, dass es sich um die beste handelt, werden anderen Bedingungen gewählt. Dies sind zum Beispiel oft eine maximale Anzahl an CPU Zyklen (Zeit), eine bestimmte Anzahl an Generationen oder es kann auch abgebrochen werden, wenn die Verbesserung der Eignung der besten Lösungsinstanz für eine bestimmte Zeit unter einem Schwellwert liegt.

De Medeiros et al. beschreiben einen Ansatz wie ein evolutionärer Algorithmus für die Process-Discovery aussehen könnte [MWA07]. Dazu verwenden sie ein eigenes Prozessmodell, welches sie „Kausal-Matrix“ nennen. Dieses wurde für die Ausführung der evolutionären Operationen optimiert. Für das Ergebnis lässt sich eine „Kausal-Matrix“ in ein Petri-Netz umwandeln. Für die **Initialisierung** nutzen sie pseudo-zufällige Modelle, welche mit Hilfe von Heuristiken optimiert sind. Für die **Bewertung** eines Modells werden nur die Kriterien **Eignung** und **Genauigkeit** verwendet (siehe Unterabschnitt 2.6.1). Die Auswahl der Eltern wird durch ein Auswahlverfahren implementiert, bei dem sie das am besten bewertete Modell aus 5 zufällig gewählten Lösungsinstanzen nehmen. Für das **Kreuzen** und die **Mutation** schlagen sie Algorithmen vor, deren Umfang größer ist, aber im Essentiellen vorhandene Instanzen nutzen, um neue zu kreieren. Für die nächste Generation werden die Elite (nur die besten Instanzen) der alten plus die gesamten neuen Instanzen gewählt. Die **Abbruchbedingung** wurde über eine maximale Anzahl an Generation n definiert, wobei auch abgebrochen wird, wenn $\frac{n}{2}$ Generationen keine Verbesserung der Lösungsinstanzen festgestellt wurde. Dieser Algorithmus ist vor allem deshalb interessant, da er (wegen dem zugrunde liegenden Modell) Zusammenhänge wie nicht lokale Abhängigkeiten modellieren kann [MWA07]. Nachteilig sind die teilweise sehr langen Laufzeiten.

Process-Mining-Framework (ProM)

Das Process-Mining-Framework [DMV+05] (kurz ProM) ist ein Softwaretool, in dem viele Erkenntnisse aus dem Bereich des Process-Mining vereint wurden. Gleichzeitig ist es auch das einzige System an dem so viele der Algorithmen des Process-Mining implementiert wurden. Es biete eine Plattform für alle Teilbereiche dieser Wissenschaft. Auch der α-Algorithmus und eine evolutionärer sind darin vorhanden. Daher ist es am weitesten verbreitete Werkzeug in dieser Disziplin.

2.7. Visualisierung von Graphen

Ein gerichteter ungewichteter Graph wird in der Mathematik als ein Tupel (V, E) definiert. Dabei ist V die Menge der Knoten und $E \subseteq V \times V$ die Menge der gerichteten Kanten. Die häufigste und intuitiv verständliche Darstellung von Graphen ist das Knoten-Kanten-Diagramm. Dabei werden oft Kreise (oder wie bei Petri-Netzen allgemein geometrische Figuren) zur Darstellung der Knoten verwendet. Die Kanten werden bei gerichteten Graphen als Pfeile zwischen den Knoten dargestellt.

Ein grundlegendes Problem ist die Positionierung der Knoten um eine übersichtliche und verständliche Darstellung zu erhalten. Fruchtermann und Reingold [FR91] beschreiben die folgenden Kriterien, die ein „ästhetisches“ Knoten-Kanten Diagramm erfüllen sollte, das in einem gegeben Ausschnitt gezeichnet werden soll:

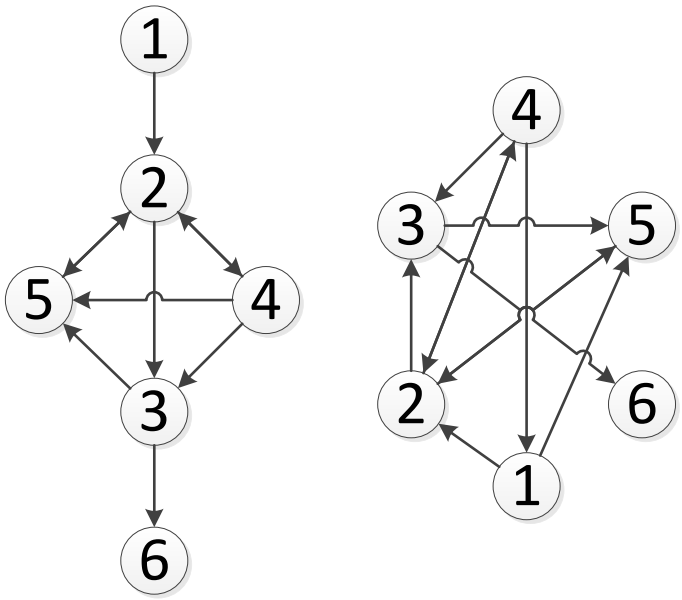


Abbildung 2.16.: Darstellung zweier Knoten-Kanten-Diagramme mit dem gleichen zugrunde liegendem Graphen.

1. Gleichmäßige Verteilung der Knoten im Bildausschnitt.
2. Minimierung der Kreuzungen von Kanten.
3. Einheitliche Längen der Kanten.
4. Darstellung von gegebener Symmetrie des Graphen.
5. Grenzen des Bildausschnittes nicht verletzen.

Die Bedeutung einer guten Verteilung der Knoten eines Knoten-Kanten-Diagrammes ist in Abbildung 2.16 zu sehen. Dabei stellen beide Diagramme den gleichen Graphen dar, da das linke Diagramm den eingeführten Kriterien besser entspricht ist die Struktur des Graphen genauer zu sehen und besser verständlich.

2.7.1. Kräftebasiertes Layout

Eine Möglichkeit das oben genante Problem zu lösen ist die Verwendung von physikalischen Prinzipien. In [FR91] wird ein Ansatz beschrieben, der vom Grundgedanken übernommen wird.

Die zugrunde liegende Vorstellung ist folgende: Die einzelnen Knoten des Graphen seien alle elektrostatisch geladene Punktladungen, z. B. metallene Kugeln. Die dadurch entstehen Kräfte sorgen für eine gleichmäßige Abstoßung der Knoten untereinander. Die Kanten zwischen den Knoten sind in diesem Modell Federn, welche den Graphen wiederum zusammenhalten. Der Gedanke ist nun dies mit physikalischen Gesetzen zu simulieren, um die gleichmäßige Verteilung und Symmetrie der Strukturen zu optimieren. Dazu werden die folgenden Konzepte verwendet:

Hookesches Gesetz

Das hookesche Gesetz besagt, dass bei der Ausdehnung einer Feder, falls diese nicht zu weit ist, die entstehende entgegenwirkende Kraft proportional zur Länge der Ausdehnung ist [wik15b].

$$F = D \cdot \Delta l$$

Für das kräftebasierte Layout hat das zur Folge, dass für die Kanten (entweder einzeln oder allgemein für alle) die Länge der Feder und deren Stärke spezifiziert werden müssen. Denn Δl lässt sich dann aus der Differenz des Abstandes der Endknoten und der Länge der Feder berechnen. Aus der Stärke folgt direkt oder indirekt die Proportionalitätskonstante D.

Coulombsches Gesetz

Die andere, anstoßende Kraft entsteht aus dem coulombschen Gesetz. Dieses besagt, dass die Kraft zwischen zweier elektrostatisch geladener Punktladungen proportional zu jeweils den beiden Ladungen und invers proportional zum Quadrat des Abstandes ist [wik15a].

$$F = \frac{1}{4\pi\epsilon_0}\frac{q_1 \cdot q_2}{r^2}$$

Die Konstante ist dabei für die Layoutsimulation nicht so interessant. Angegeben werden müssen daher nur die Ladungen der einzelnen Knoten.

Barnes-Hut-Approximation

Die Implementierung aus [d3215] nutzt die Barnes-Hut-Approximation, um die Berechnung der abstoßenden Kräfte zu beschleunigen. Da normalerweise bei einem solchen Typ von Simulation der Aufwand in $\mathcal{O}(n^2)$ in der Größe der Elemente (hier Knoten) wächst, wurde J. Barnes und P. Hut eine Annäherung entwickelt [BH86]. Diese basiert auf der Idee, dass ab einem gewissen Abstand die Kräfte nicht mehr so genau berechnet werden müssen, da mit großem Abstand der Einfluss schnell kleiner wird.

Der Algorithmus arbeitet dabei auf einem Quadtree, welcher den zweidimensionalen Raum und die Elemente unterteilt (in einem dreidimensionalen wäre es ein Octtree). Anstatt für jedes Element die Kräfte mit allen anderen Elementen zu berechnen, wird dies nur für Elemente in nächster Nähe getan. Für Quadrate, welche eine größere Entfernung besitzen, wird die Kraft zu einem virtuellen, für die Elemente des Quadrates stellvertretenden, Element berechnet. Diese Vereinfachung sorgt für eine drastische Verbesserung der Zeitkomplexität auf $\mathcal{O}(n \log n)$.

Zusammenfassung und Beispiele

Durch die Kombination dieser Konzepte entsteht ein Algorithmus, der eine gute Performanz hat. Der Algorithmus ist ein iterativer Prozess, bei dem initial die Knoten zufällig verteilt werden. In jeder Iteration werden die anziehenden und abstoßenden Kräfte berechnet und anschließende die Knoten entsprechend verschoben. Damit der Prozess auch sicher in einen stabilen Zustand kommt, wird ein Abschwächungsfaktor integriert. Dieser wird nach jeder Iteration verringert, um somit die Kräfte immer geringer werden zu lassen und das Layout in einen stabilen Zustand zu geleiten. Die Abbildung 2.17 und Abbildung 2.18 zeigen zwei Beispiele, welche mit dem D3-Framework erstellt wurden und Ergebnisse des kräftebasierten Layouts zeigen.

2.8. Verwandte Arbeiten

Dieser Abschnitt verweist auf verwandte Arbeiten, die zu Teilen in Arbeit verwendete Konzepte besprechen. Dabei existieren zwei Typen an relevanten Arbeiten, die einen, welche Problemstellungen des Process-Mining erörtern und die anderen, welche Visualisierungen, gerade von Graphen und Modellnotationen, besprechen.

In [ARW+07] untersuchen van der Aalst et. al die Anwendung des Business-Process-Mining anhand echter Daten. Diese Daten stammen aus den Logs der Prozess einer Abteilung der staatlichen Tiefbaubehörde. Dabei verwenden sie für die Erstellung der Prozessmodelle einen heuristischen Ansatz [WA03, VDA11] im Gegensatz zu dieser Arbeit, die sich auf den α-

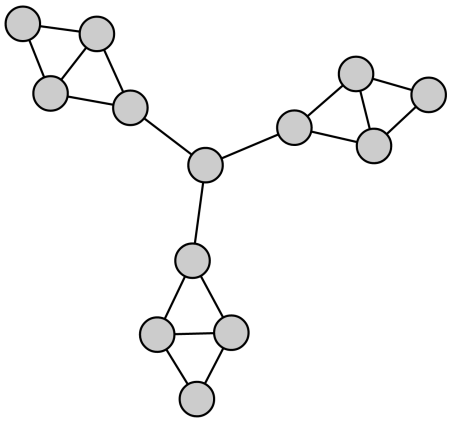


Abbildung 2.17.: Ein simpler Graph mit Symmetrie, die gut durch das kräftebasierte Layout zum Vorschein kommt. [mbo12]

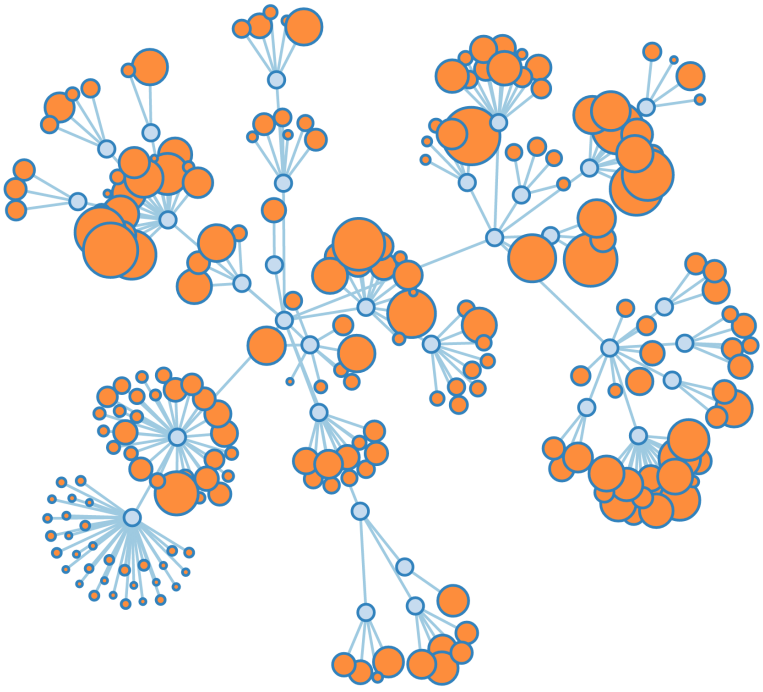


Abbildung 2.18.: Knoten-Kanten-Diagramm eines Baumes, dessen Struktur gleichmäßig verteilt wird, durch das kräftebasierte Layout. [mbo11]

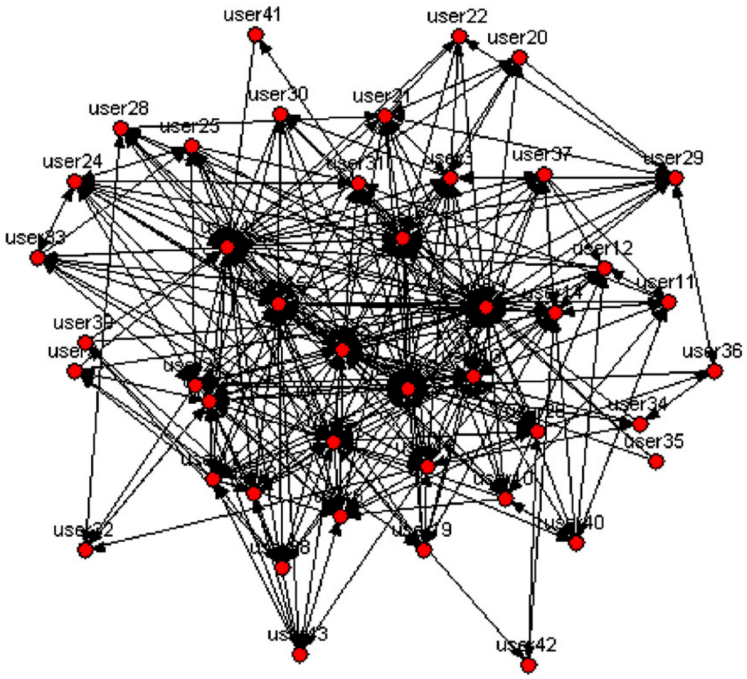


Abbildung 2.19.: Erstellter Kommunikationsgraph aus [ARW+07].

[AWM04] und den evolutionären Algorithmus beschränkt [MWA07]. Dabei kommen sie zu dem Ergebnis, dass in der Praxis erwartete Probleme, wie Rauschen (fehlerhafte Abläufe oder Traces), sich gut durch den heuristischen Ansatz lösen können und auch das ProM-Framework schon für praktische Aufgaben brauchbar ist. Auf eine gute visuelle Aufbereitung ihrer erstellten Ergebnisse legen sie dabei weniger Wert, wie sich gut in Abbildung 2.19 sehen lässt.

Wil van der Aalst beschreibt in [VDA13] die wachsende Bedeutung des Process-Mining für Webservices und Systeme mit serviceorientierter Architektur. Dabei untersucht er die theoretischen Problematiken die bei der Erstellung von System entstehen welche sehr auf Webservices, gerade auch mit Kommunikation über die Grenzen einer einzigen Organisation hinweg, basieren. So sei ein großes Problem, dass bei der Entwicklung die Abläufe in anderen Services nicht bewusst sind, da sie zum Teil gar nicht dokumentiert sind. Da sich der ganze Prozess aber über mehrere Systeme streckt, kann es zu unvorhergesehen Problemen, z. B. Deadlocks, kommen. Ein Deadlock ist eine Markierung (welche nicht die Endzustand-Markierung ist), bei dem keine aktive Transition mehr existiert, d. h. keine Aktionen sind mehr möglich, obwohl der Prozess noch nicht geendet hat. Das Netz in Abbildung 2.20 kommt in einen solchen Zustand nach der Ausführung von $\langle po, ro, sp, sr \rangle$. Dies ist nur zu erkennen, wenn das gesamte Netz bekannt ist. Diese Beschreibung ist nur theoretischer Natur und verdeutlicht die Bedeutung des Prozess-Mining für Webservices und welche Herausforderungen noch zu bewältigen sind. Das in der vorliegenden Arbeit verwendete

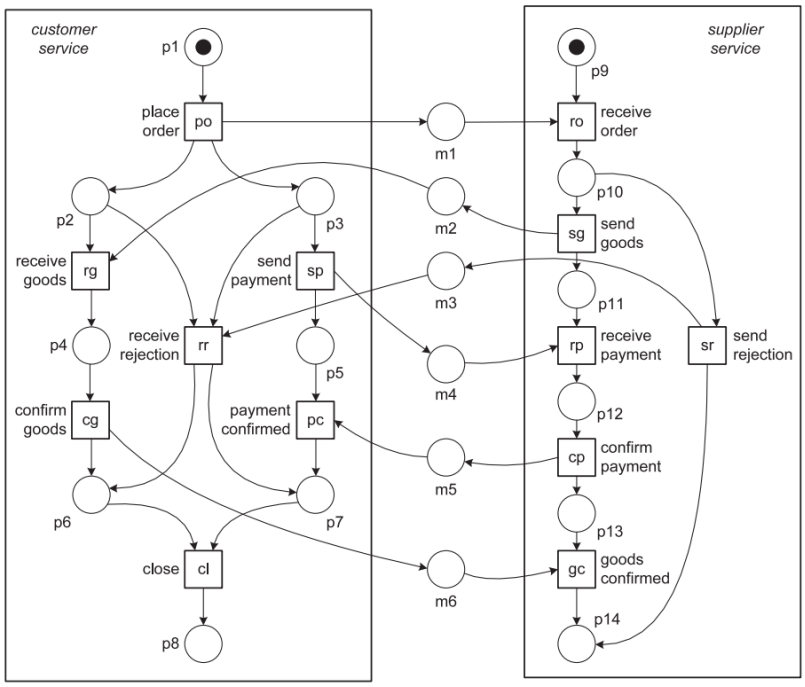


Abbildung 2.20.: Petri-Netz von zwei Webservices mit einem enthalten Deadlock [VDA13]

System ist auch von serviceorientierter Natur, hat jedoch nicht alle diese Probleme, da das System nicht über die Organisationsgrenzen hinaus geht.

Eine dazu passende Arbeit ist [AV08]. In dieser wird die Anwendung von Process-Mining auf ein System mit einer IBM Websphere Umgebung untersucht. Van der Aalst und Verbeek betrachten dabei welche unterstützenden System Websphere für das Process-Mining mitbringt und wie sich damit Daten für das ProM-Framework gewinnen lassen. Dabei gehe sie nur auch die technischen Einzelheiten von Websphere ein und beschreiben keinen Fall, den sie mit dem System analysiert haben. Sie beschreiben Ansätze, wie solche serviceorientierten Softwaresysteme mit dem Process-Mining verbunden werden können. Im Gegensatz zu dieser Arbeit betrachten sie dabei keine realen Daten und überprüfen damit nicht die praktische Anwendbarkeit. Auch unterscheidet sich, dass diese Arbeit das ProM-Framework nur für die Mining-Algorithmen verwendet und Visualisierung und Analyse selber übernimmt und erweitert.

Cook und Wolf gehen einen etwas anderen Ansatz in [CW98]. Ihr Ziel ist dasselbe wie auch in dieser Arbeit, die Erstellung von Prozess Modellen für Software mit Hilfe von Eventdaten. Dabei sehen sie auch die Bedeutung der Einfachheit und Automatisierung der Erstellung solcher Modelle, um die Wartung von Software zu vereinfachen. Dabei nutzen sie nicht die Techniken des Process-Mining [VDA11], sondern sie verwenden Techniken aus dem Machine Learning wie neuronale Netzwerke und Markov-Modelle. Sie verwenden auch ähnliche Qualitätsindikatoren (siehe Unterabschnitt 2.6.1) um die erstellten Modelle zu bewerten, da dies für Algorithmen aus der künstlichen Intelligenz nötig ist.

Die bisherigen Arbeiten betrachten alle das Herausfinden des Prozessflusses und weniger die Visualisierung der entstandenen Modelle. Eine Arbeit die kräftebasierte Layouts für die Visualisierung von formalen Notationen benutzt ist [Dwy01]. Dwyer untersucht dabei wie kräftebasierte Layouts für die Visualisierung von UML-Klassendiagrammen verwendet werden können. Dabei werden diese im Dreidimensionalen modelliert und angezeigt. Dabei bewertet er den kräftebasierten Ansatz als gelungen und eine Analyse unterstützend.

3. Konzept

Dieses Kapitel erörtert das in dieser Arbeit zu untersuchende Konzept.

Es wurde ein Konzept zum Mining und zur Visualisieren der Prozesse entworfen. Dazu sind die in Abbildung 3.1 als Verarbeitungs-Pipeline dargestellten Verarbeitungsschritte notwendig. Die Logdaten sind in einem Datenspeicher vorhanden. Die darin vorhandenen Logeinträge werden gefiltert und in das XES-Format umgewandelt (siehe Vorverarbeitung der Logdaten). Danach werden aus den erstellten Traces Prozessmodelle erstellt (siehe Process-Mining). Und schließlich werden die Modelle und Traces für eine gute Analyse passend visualisiert (Visualisierung und visuelle Analyse).

Zusätzlich wurde ein Konzept entwickelt, das die Hierarchie der Serviceaufrufe zur Optimierung der Verarbeitung nutzt (Bezeichnerkonvention der Hierarchie).

3.1. Vorverarbeitung der Logdaten

Die einzelnen Logeinträge, die direkt aus der Datenquelle stammen, lassen sich nicht direkt in einem Process-Mining-Tools verwenden, denn zum einen müssen die Daten ein vom Process-Mining-Framework (ProM) [DMV+05] lesbares Format gebracht werden (hier XES), zum anderen müssen spezielle Einträge als relevant eingestuft werden und auf entsprechende Events abgebildet werden. Bei dem vorliegenden System werden nur der Beginn der Ausführung eines Services und die Beendigung der Ausführung auf Events abgebildet. Alle anderen Einträge werden nicht betrachtet, könnten jedoch teilweise für weitere Verbesserungen der Process-Discovery oder andere Arten des Process-Mining durchaus relevant sein.

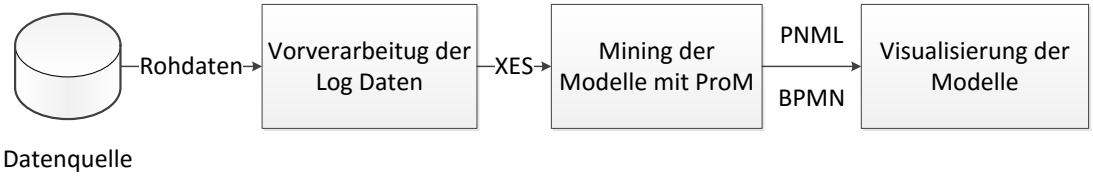


Abbildung 3.1.: Darstellung der Verarbeitungsschritte in einer Pipeline.

Die Einträge beinhalten Start und Ende der Services und deren Zuordnung zu einem Vorgang. Was bei den vorliegenden Daten leider fehlt, ist die Zuordnung der Vorgänge zu einem Prozess oder einem Use-Case. Eine Zuordnung der Traces zu einem Prozess ist aber unbedingt nötig, um ein verwertbares Modell zu bekommen, gerade bei einem System mit einer großen Zahl an Services und Prozessen. Daher wurde die folgende Heuristik für die Zuordnung zu einem Prozess gewählt: Um die Traces für ein Prozessmodell zu bekommen, wird ein bestimmter Service gewählt. Dieser dient als Filter, um nur Traces zu behalten, welche bei der Ausführung diesen Service verwenden. Dabei ist zu beachten, dass einige Services nicht geeignet sind, da sie von einer Vielzahl an anderen Services verwendet werden. Viele Prozesse oder Anwendungsfälle haben aber einen Service, der ausschließlich von diesen verwendet wird und die Traces somit charakterisieren können.

Nach dem ausführlichen Filtern der Einträge werden die Traces in das XES-Format, dabei werden Traces über die Tracking-ID definiert. Events erhalten ihre relative Reihenfolge über die Sortierung nach ihrem Zeitstempel, an dem der Eintrag in den Log eingetragen wurde. Der Name ergibt sich über den Service. Start und Ende eines Service werden auf die entsprechende Lebenszyklus Definitionen in XES abgebildet.

3.2. Process-Mining

Der nächste Verarbeitungsschritt ist das Erstellen geeignet Modelle aus den Logdaten. Die Logdaten aus der Datenquelle liegen nach dem vorherigen Verarbeitungsschritt als XES vor. Dazu werden zwei Algorithmen verwendet, zum einen der α-Algorithmus (siehe Unterabschnitt 2.6.2) und zum anderen eine Implementierung des evolutionären Ansatzes (siehe Unterabschnitt 2.6.4). Dabei sollen die Algorithmen nicht selber implementiert werden, sondern das vorhandene ProM-Framework genutzt werden, um aus den vorliegenden Logdaten Modelle zu generieren. Deshalb ist auch nötig, dass die Ausgabe des ersten Verarbeitungsschrittes in einem standardisierten Format ist. Der Aufruf des ProM-Frameworks sollte ohne Hilfe des Nutzers geschehen, über die Nutzung einer Kommandozeilen-Schnittelle. Nach der Terminierung der Algorithmen liegen Petri-Netz oder BPMN in einem standardisierten Format vor, welches wiederum vom zu entwickelnden System gelesen und weiter verarbeitet wird. Für den α-Algorithmus ist es kein Problem ihn ohne Nutzereingaben auszuführen, da er keine weiteren Eingaben benötigt außer den Logdaten. Für einen evolutionären Algorithmus ist dies nicht der Fall, dieser besitzt eine Vielzahl an Parametern, wie Größe der Population, Koeffizienten der Bewertungsfunktion, etc.. Um die Komplexität des Systems aber nicht unnötig zu vergrößern, werden diese manuell auf einen Standardwert gesetzt. Dazu werden empfohlene Standardwerte verwendet.

3.3. Visualisierung und visuelle Analyse

Der letzte Verarbeitungsschritt ist die visuelle Aufbereitung der Modelle (Petri-Netze und BPMN). Dabei ist zum einen eine übersichtliche Darstellung der Graphen notwendig, zum anderen soll auch noch eine visuelle Unterstützung implementiert werden für das Verständnis, wie aus den gegebenen Logdaten das Modell entstanden ist.

Für die übersichtliche Darstellung der Graphen können ein kräftebasiertes Layout oder andere Techniken verwendet werden. Damit werden die Knoten gleichmäßig verteilt mit wenigen Kantenüberschneidungen. Bei dem Layout ist zusätzlich zu beachten, dass die Rotation des Graphen nicht vernachlässigt wird. Im Allgemeinen wird bei einem kräftebasierten Layout die Rotation nicht in die Optimierung miteinbezogen. Die Übersichtlichkeit eines Prozessmodelles ist aber nicht rotationsinvariant.

Zur Unterstützung des Verständnisses der erstellten Graphen und zur Darstellen der Plausibilität soll die Visualisierung eine Möglichkeit bieten, die zugrunde liegenden Traces anzuzeigen. Dazu soll es möglich sein für einen ausgewählten Trace den Verlauf der Ausführung im Modell anzuzeigen. Dies versichert dem Nutzer, dass ein aufgenommener Ablauf auch tatsächlich mit dem Modell ausführbar ist. Außerdem bieten diese beispielhaften Ausführungen einen guten Anhaltspunkt um das Modell schneller zu verstehen, da der Nutzer eine Ausführungsmöglichkeit sieht und auch schneller erkennt, welche anderen Möglichkeiten noch existieren. Zusätzlich soll noch die Möglichkeit bestehen, die Traces nach Transitionen Filtern zu können um die Anzahl der zugrunde liegenden Traces zu erkennen. Damit kann ein Nutzer die Signifikanz eines Teilmodells abschätzen.

3.4. Hierarchisierung der Traces und der Modelle

Aufgrund der reellen Anwendung des Systems können einzelne Traces schnell sehr lang werden und damit auch größere Modelle zur Folge haben. Dies beeinflusst nicht nur die Visualisierung, sondern auch das Mining der Modelle. Obwohl der α-Algorithmus keine Probleme mit größeren Traces in Bezug auf seine Laufzeit hat, schränken die Probleme des Algorithmus den Nutzen in der Praxis ein (siehe Unterabschnitt 2.6.3). Der evolutionäre Algorithmus hingegen skaliert schlecht in der Größe des Logs und in der Größe des Modelles. Daher wurde ein Ansatz entworfen, der zwar nicht das Problem löst, aber einige Verbesserungen mit sich bringt.
Die grundlegende Idee ist die folgende: In einem Softwaresystem sind viele Aufrufe nicht auf der obersten Ebenen, sondern Unteraufrufe eines weiteren Services. Dieser Zusammenhang soll genutzt werden, indem jedem Event-Eintrag sein aufrufender Service mitgegeben wird.

3.4.1. Bezeichnerkonvention der Hierarchie

Um ein Event oder einen Task in seinen Kontext einzuordnen, soll es mit der globalen Aufrufhierarchie bezeichnet werden. Dazu wird der Stapel der übergeordneten Task, jeweils mit einem „::“ getrennt, vor den Bezeichner geschrieben. Den Zustandsübergang des Lebenszyklus (in diesem Fall nur „start“ oder „complete“) wird getrennt mit einem „+“ nach dem Bezeichner angefügt. Die Konvention zur Integration des Lebenszyklus in den Bezeichner wurde aus ProM übernommen. Zur Erklärung sei folgender Beispiel-Trace gegeben.

$$\sigma = \langle \textbf{a}\text{+start}, \text{a::}\textbf{b}\text{+start}, \text{a::b::}\textbf{c}\text{+start}, \text{a::b::}\textbf{c}\text{+complete}, \text{a::}\textbf{b}\text{+complete}, \\ \textbf{a}\text{+complete}, \textbf{d}\text{+start}, \text{d::}\textbf{b}\text{+start}, \text{d::}\textbf{b}\text{+complete}, \textbf{d}\text{+complete} \rangle$$

Wie oben zu sehen ist, wird b, wenn es im Kontext von a verwendet wird, mit a::b bezeichnet und entsprechend im Kontext von d mit d::b. Damit werden diese beiden Events mit einem anderen Bezeichner versehen und werden von dem α-Algorithmus nicht als dasselbe Event betrachtet. Wie oben am Beispiel von c zu sehen ist, werden weiter verschachtelte Aufrufe mit der gesamten Hierarchie versehen. In diesem Beispiel wird c mit a::b::c bezeichnet, da es im Kontext von b aufgerufen wurde, welches wiederum dem Kontext von a entspringt.

3.4.2. Vorteile

Auch die entstehenden Modelle sollen zuerst diese Bezeichnung behalten, d. h. die globalen Bezeichner in der Beschreibung verwenden und später in der Visualisierung nur noch die lokalen Bezeichner verwenden, um die Übersichtlichkeit zu verbessern. Daraus ergeben sich insgesamt beim Erstellen und der Visualisierung die folgenden Vorteile:

- Die Genauigkeit der vom α-Algorithmus erstellten Petri-Netze wird verbessert, da das Problem der nicht lokalen Abhängigkeiten verhindert wird (siehe Abschnitt 2.6.3).
- Anstatt ein Modell für den globalen Prozess zu erstellen, kann für jeden Subprozess ein eigenes Modell erstellt werden. Dies lässt sich so weit optimieren, dass für jedes Präfix (Teilbezeichner vor dem letzten „::“) ein flaches Modell erstellt wird. Dies bedeutet, dass keine Events betrachtet werden, welche eine Ebene tiefer oder höher (bis auf das Start und Ende Event das dem Präfix entspricht). Damit für jeden Subprozess ein Modell erstellt in welchem nur die direkt untergeordneten Aufrufe verwendet werden. Durch das rekursive Einsetzen jedes Modells der Subprozesse in das globale Modell (das auch nur mit globalen Events erstellt wurde) lässt sich das Modell des gesamten Prozesses erstellen.
- Durch das Übernehmen der hierarchischen Bezeichner in das Modell ist eindeutig, welche Task anderen untergeordnet sind. Dies kann somit ausgenützt werden, um die einem Task untergeordneten Subtask ein- und ausblenden zu können. Dies sorgt für eine übersichtlichere Darstellung und hilft somit dem Verständnis des Modells.

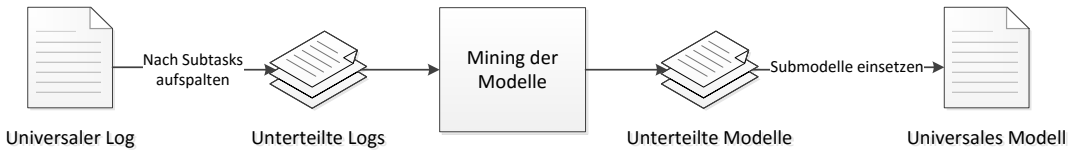


Abbildung 3.2.: Verarbeitungsablauf, welcher die Hierarchie des Logs nutzt.

- Auch die räumliche Verteilung der Knoten des Graphen kann mit der Hierarchie optimiert werden, indem das Modell zuerst nur die globalen Elemente beinhaltet und der Rest ausgeblendet ist. Damit wird die Anzahl der Knoten reduziert und eine Optimierung des Layouts liefert bessere Ergebnisse.

3.4.3. Nutzung der Hierarchie in der Process-Discovery

Wenn Subevents im Log als solche annotiert sind, ist es sinnvoll für jeden Subprozess das Modell getrennt zu erstellen. Dazu wird, wie in Abbildung 3.2 abgebildet, der gesamte Log in kleinere Logs unterteilt, welche jeweils nur die Events beinhalten, die direkt zu einem bestimmten Subprozess gehören. Das sind zum einen das Start- und Ende-Event eines Subprozesses und zum anderen direkt in diesem Kontext ausgeführte Start-Events (es ist nur ein Event nötig welches repräsentativ für den ausgeführten Subprozess steht). Außerdem wird ein globaler Log erstellt, der nur die Einträge enthält, die nicht im Kontext eines anderen Service aufgerufen wurden. Nach dem Erstellen der Modelle für jeden Log einzelnen, werden die Modelle für die Subprozesse an entsprechender Stelle rekursiv in das aus dem globalen Log entstandenen Modell eingesetzt. Dies geschieht, indem die repräsentative Transition entfernt wird und das Ziel aller eingehenden Kanten auf die Start-Transition des Subnetzes geändert wird. Genauso wird der Ursprung der ausgehenden Kanten auf die Ende-Transition des Subnetzes geändert.

Dies soll an einem Beispiel verdeutlicht werden.

$$L = [\langle \mathbf{a}\text{+start}, \text{a::}\mathbf{b}\text{+start}, \text{a::}\mathbf{b}\text{+complete}, \mathbf{a}\text{+complete}, \mathbf{d}\text{+start}, \mathbf{d}\text{+complete}\rangle,$$
$$\langle \mathbf{a}\text{+start}, \text{::}\mathbf{c}\text{+start}, \text{a::}\mathbf{c}\text{+complete}, \mathbf{a}\text{+complete}, \mathbf{d}\text{+start}, \mathbf{d}\text{+complete}\rangle]$$

Daraus entsteht folgender globaler Log:

$$L_{global} = [\langle \mathbf{a}\text{+start}, \mathbf{d}\text{+start}\rangle,$$
$$\langle \mathbf{a}\text{+start}, \mathbf{d}\text{+start}\rangle]$$

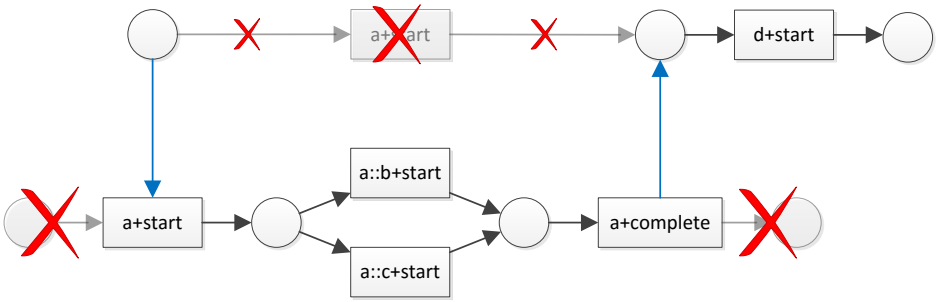


Abbildung 3.3.: Globales Petri-Netz und Petri-Netz für Subprozess a. Die Transition a+start wird durch sein Subprozess-Modell ersetzt.

Und folgende lokale Logs für jeden einzelnen Subprozess:

$$
\begin{aligned}
L_a &= [\langle \textbf{a}\text{+start}, \text{a::}\textbf{b}\text{+start}, \textbf{a}\text{+complete}\rangle, \\
&\quad \langle \textbf{a}\text{+start}, \text{a::}\textbf{c}\text{+start}, \textbf{a}\text{+complete}\rangle] \\
L_{a::b} &= [\langle \text{a::}\textbf{b}\text{+start}, \text{a::}\textbf{b}\text{+complete}\rangle] \\
L_{a::c} &= [\langle \text{a::}\textbf{c}\text{+start}, \text{a::}\textbf{c}\text{+complete}\rangle] \\
L_d &= [\langle \textbf{d}\text{+start}, \textbf{d}\text{+complete}\rangle]
\end{aligned}
$$

In Abbildung 3.3 sind Petri-Netze für die Logs L_{global} und L_a abgebildet. Außerdem zeigt die Abbildung das Ersetzen der a+start Transition durch das Petri-Netz des Subprozesses von a.

Für den α-Algorithmus hat das keine großen Vorteile. Für den evolutionären Algorithmus ist der Vorteil hingegen immens, da die Komplexität linear in der Größe des Modells und der Größe des Logs wächst und beides durch den Ansatz reduziert wird.

4. Implementierung

Diese Kapitel beschreibt die Implementierung des untersuchten prototypischen Systems. Dieses greift die Gedanken aus Kapitel 3 auf.

4.1. Ausgangssituation und Daten

Diese Arbeit nutzt reale Daten aus einem komplexen Anwendungssystem eines großen deutschen Autoherstellers. Sie soll eine mögliche Verbesserung des Verständnisses und der Wartbarkeit des Systemes untersuchen und Process-Mining auf komplexen Anwendungssystemen anwenden. Die Nutzung der realen Daten bietet zum einen bessere Beurteilung des Konzeptes auf Praxistauglichkeit, zum anderen müssen keine Testdaten erstellt werden. Ziel der Untersuchung ist es, die Prozesse, welche in dem System ablaufen, in einem Prozessflussmodell darzustellen. Diese Darstellung sollte ein besseres und tieferes Verständnis des Kommunikationsflusses bieten. Vor allem sollten diese Modelle auch eine weitreichende Verbesserung bestehender Systeme zur Analyse bieten. Eine besondere Beachtung soll auch die Visualisierung der erstellten Modelle und eine unterstützende Darstellung der Daten in dem Modell bekommen, die dazu dienen sollen dem Anwender eine bessere Nachvollziehbarkeit des Erstellungsprozesses zu bieten.

4.1.1. Das Anwendungssystem

Das gegebene Anwendungssystem besteht im ganzen aus ca. 30 autonomen Systemen, welche nach der serviceorientierten Architektur Dienste anbieten. Anfragen von außen, z. B. im Auto oder auf einer Webseite ausgelöst, können zum Teil die Mitwirkung von vielen Systemen zur Folge haben. Durch die hohe Komplexität und einer relativ häufigen Zahl an Deployments, gibt es oft keine Dokumentation des Prozessflusses oder diese weicht von der Realität ab. Das Process-Mining bietet viele Methodiken, um hier Abhilfe zu schaffen. Diese Arbeit wird sich aus Sicht der Process-Discovery (siehe Abschnitt 2.6) an das Problem annähern. D. h. es soll versucht werden ein System zu entwickeln, welches für die ablaufenden Prozesse möglichst passende Modelle erstellt. Andere Sichten wie Kommunikationsgraphen, welche auch oft beim Process-Mining erstellt werden [VDA11], sind nicht Teil dieser Arbeit sollten jedoch in einem praktischen System nicht vernachlässigt werden.

Um ein Modell des Prozessflusses entdecken zu können, müssen Prozessabläufe ausführlich dokumentiert werden. In diesem Fall ist dies gegeben durch Logeinträge, die bei jedem Aufruf und jedem Abschluss eines Services geschrieben werden, diese sind durch die Texte „Application enter“ und „Application exit“ identifiziert. Der aufgerufene Service und das System sind in eigenen Feldern gegeben. Damit haben diese Logs nicht höchste Qualität nach [VDAAM+12], aber sind noch gut für das Process-Mining geeignet, wenn eine geeignete Vorverarbeitung stattfindet. Diese zwei Typen von Einträgen sollen die Einzigen sein, die von dem System betrachtet werden, alle anderen Einträge werden ignoriert.

4.1.2. Elasticsearch

Für die Logdaten der Systeme existiert eine Elasticsearch-Instanz, welche diese speichert und indiziert. Das bestehende System zur Analyse des Prozessflusses basiert auf einer Kibana-Instanz die Elasticsearch als Datenquelle nutzt. Kibana bietet einen verbesserten Zugriff auf die rohen Daten aus Elasticsearch. Es bietet einfach zu erstellende Diagramme, welche aus den Daten berechnete Metriken darstellen. Dies kann zum Beispiel die Anzahl von Aufrufen bestimmter Services sein. Darüber hinaus bietet Kibana aber kein großes Data-Mining, sondern nur eine bessere Möglichkeit auf die Daten aus Elasticsearch zuzugreifen und diese zu verstehen. Kibana ist die aktuell genutzte Analysemethode der gegebenen Daten. Gegen diese wird der Ansatz verglichen und auf eine Verbesserung der Analysemöglichkeiten untersucht. Damit die Logdaten von den System in das Elasticsearch gelangen wird Logstash auf den Instanzen ausgeführt und liefert die Logdaten an den Indexer des Elasticsearch weiter.

4.1.3. Tracking-ID

Ein allgemeines Problem in Logs ist die Korrelation der Einträge, die Zuordnung zu einem Prozess und zu einem Vorgang. Wenn zum Beispiel in einem Log ein Fehler auftaucht, so sind zur Findung der Ursache die zum gleichen Vorgang gehörenden Einträge interessant. Diese befinden sich nicht nur im gleichen Log, sondern auch in anderen Logs, welche zum Teil sich auf mehreren verschiedenen Systemen befinden. Dabei löst das oben beschriebene System mit Logstash und Elasticsearch das Problem, die Daten an einen zentralen Ort zu sammeln. Die Sortierung der Einträge nach Vorgängen ist damit jedoch nicht gelöst.
Das Problem der Korrelation all dieser Einträge wurde in dem vorliegenden System folgendermaßen gelöst: Zu Beginn der Abarbeitung, wenn eine Anfrage zum ersten Mal das System betritt, wird eine eindeutige Identifikationsnummer erstellt. Diese wird bis zur vollständigen Abarbeitung der Anfrage von System zu System mitgeführt und auch allen Logeinträgen hinzugefügt. Somit können alle zu einer Anfrage gehörenden Einträge bestimmt werden. Diese **Tracking-ID** ermöglicht erst das Process-Mining, da damit die Traces eines Vorganges erstellt werden können, in dem nach einer bestimmten Tracking-ID gefiltert wird.

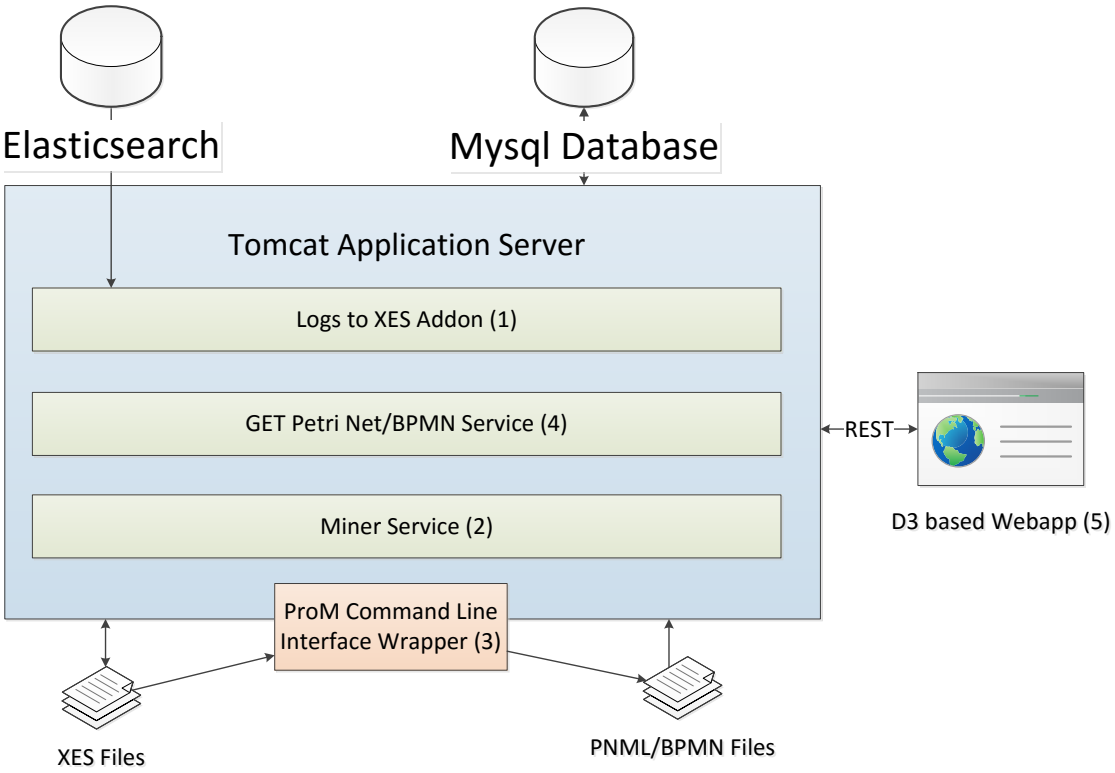


Abbildung 4.1.: Architektur des Systems.

4.2. Architektur

Die Systemarchitektur ist abgebildet in Abbildung 4.1. Die Hauptanwendung läuft in einem Tomcat-Applikation-Server, es handelt sich um eine Java-RS J2EE Anwendung. Die Visualisierung wird ausgelagert eine Webanwendung, um das Data Driven Documents (D3) Framework verwenden zu können. Über die Java Persistence API wird eine MySQL-Datenbank an das System angebunden, um die anfallenden Daten persistent speichern zu können. Die erzeugten Dateien (Logs und Modelle) werden im Dateisystem gehalten und darüber auch dem Process-Mining-Framework (ProM), beim Aufruf über die Kommando Zeile, übergeben. Das System bietet eine Webservice Schnittstelle nach dem Representational State Transfer (REST) Paradigma, dessen Endpunkte von der Webapp aufgerufen werden. Die von der API ausgegebenen Daten sind ausschließlich im JSON-Format.

Die Quelle für die Daten stellt das Elasticsearch dar (siehe Unterabschnitt 4.1.2). Dieses bietet eine REST-API um Anfragen gegeben die Daten auszuführen und liefert diese in einem, für NoSQL typisch, JSON-Format zurück. Die Aufgabe der „Logs to XES“ (siehe Abbildung 4.1 (1)) Komponente ist die Vorverarbeitung der gegebenen Daten und die Umwandlung in das XES-Format. Diese Komponente ist die Einzige vom dem untersuchten Anwendungssystem

abhängige Komponente. Die restliche Anwendung kann auch mit vorhanden XES-Dateien verwendet werden und kann somit allgemein genutzt werden.

Der nächste Schritt der Verarbeitung wird vom „Miner Service“ (siehe Abbildung 4.1 (2)) gesteuert. Dieser bietet einen Endpunkt für die Erzeugung von Modellen aus den vorhandenen Logs (XES-Dateien). Dieser Prozess wird durch das ProM-Framework durchgeführt. Dazu existiert ein Wrapper (siehe Abbildung 4.1 (3)), welcher die Steuerung von ProM über die Kommandozeile übernimmt. Dabei kann zwischen dem α- und einem evolutionären Algorithmus gewählt werden. Als Ausgabe entstehen Petri-Netze oder BPMNs.

Die letzte Komponente (siehe Abbildung 4.1 (4)) dient der Lieferung der vorhandenen Modelle an die Webapp. Um die Verwendung zu vereinfachen, werden diese nicht in dem vorliegenden Format, die direkte Ausgabe von ProM, sondern in einem eigenen vereinfachten JSON-Format übergeben. Die Dateiformate PNML und BPMN sind beide XML-basiert und enthalten beide um einiges mehr Informationen als für Beschreibung und Darstellung des zugrunde liegenden Modells nötig ist. Deshalb ist eine Umwandlung in ein reduziertes JSON-Format sinnvoll und auch für die Verarbeitung im D3-Framework von großem Vorteil.

Das Userinterface des Systems ist als Webapp (siehe Abbildung 4.1 (5)) realisiert. Dadurch werden Datenhaltung und UI klar getrennt und für die Visualisierung können die Vorteile von D3 und die Geschwindigkeit der Render-Engine der Browser ausgenutzt werden. Die Aufgabe der Webapp ist vor allem die Visualisierung der Petri-Netze und BPMN-Modelle, aber auch eine Flussvisualisierung der Traces des Logs, aus dem das Modell entstanden ist. Indem die beobachteten Abläufe im Modell gezeigt werden, soll die Entstehung und Bedeutung des Modells besser verständlich gemacht werden.

4.3. REST-Schnittstelle

„REpresentational State Transfer“ (REST) ist eine Architektur für die Schnittstellen von Webservices [BS07]. Diese nutzt meistens den HTTP-Standard und dessen verschiedene Operationsmodi, um Ressourcen abzurufen und zu manipulieren.

Da das System in ein Backendsystem und eine Webapp ausgeteilt ist, wird die Kommunikation dieser Komponenten locker über eine REST-Schnittstelle implementiert. Die Funktionen dieser Schnittstelle werden in diesem Abschnitt genauer beschrieben. Dies dient auch der Vorstellung der Funktionalität, die das Backendsystem bietet. Dazu werden die wichtigen Endpunkte und deren Verhalten vorgestellt. Ergebnisse und Parameter sind alle im JSON-Format, mit Ausnahme der Aufrufe welche „GET“ aufgerufen werden. Bei diesen sind die Parameter in die URL codiert.

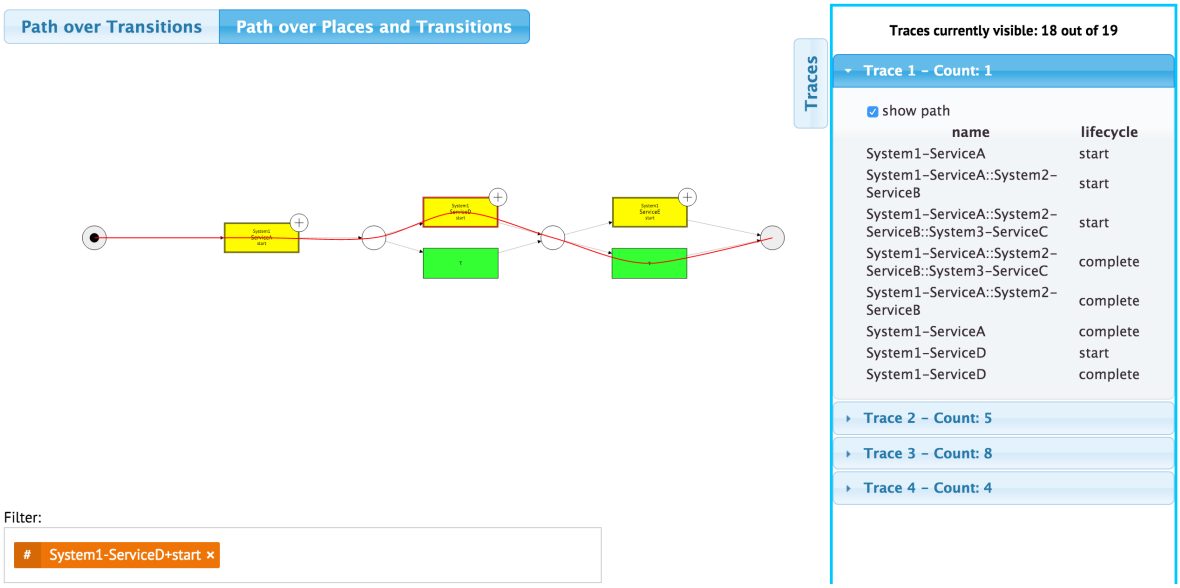


Abbildung 4.2.: Die Darstellung zeigt die Webapp mit angezeigtem Log. Im Modell wird der Pfad des Trace 1 visualisiert.

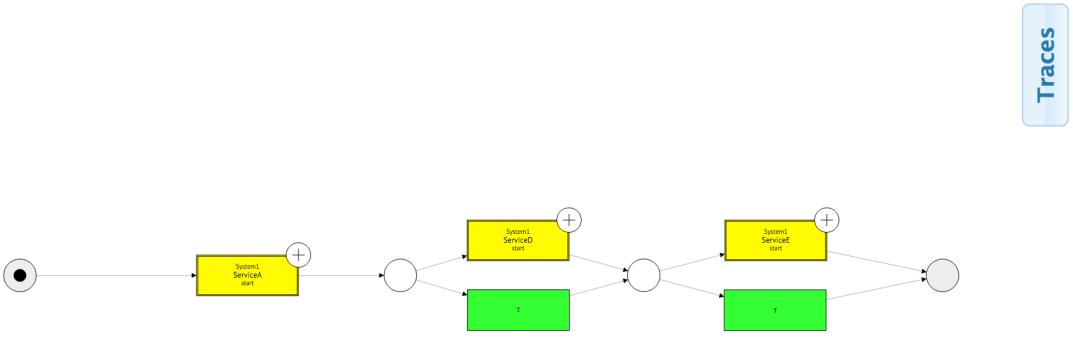


Abbildung 4.3.: Durch das Ausblenden der UI-Elemente entsteht mehr Platz für die Visualisierung und die Übersichtlichkeit wird erhöht.

4.3.1. Endpunkt zum Erstellen eines Logs

Der folgende Endpunkt bietet Zugriff auf die Komponente zur Erstellung einer XES-Datei. Beim Aufrufen der Prozedur wird asynchron der Prozess gestartet. Nach der Fertigstellung wird die XES-Datei in das System eingetragen und der Endpunkt zur Auflistung der XES-Dateien (siehe Unterabschnitt 4.3.2) liefert eine Liste, welche die erzeugte Datei beinhaltet.

Endpunkt: XES erstellen		
Beschreibung:	Endpunkt zum Starten des Prozesses zur Erstellung einer XES-Datei aus den Logdaten des Anwendungssystems.	
URL:	/createXES	
Type	POST	
Parameters		
Bezeichner:	Datentyp	Beschreibung
service	String	Der Name des Service, nach dem die Log gefiltert werden (siehe Abschnitt 3.1).
traceCount	Integer	Anzahl der Traces die erstellt werden sollen.
range	Integer	Anzahl an Tagen, die einen Zeitraum seit dem aktuellen Datum definieren. Die Daten für die zu erstellen Logs werden nur aus diesem Zeitraum gewählt.
environment	String	Definiert von welcher Umgebung (Test, Integration, Produktion, ...) das Elasticsearch verwendet werden soll, das als Quelle der Daten dient.
Ergebnis		
Bezeichner:	Datentyp	Beschreibung
info	String	Nachricht, welche das erfolgreiche Starten des Vorgangs ausgibt.

4.3.2. Endpunkte zur Verwaltung der Dateien

Das System verwaltet die Dateien von Logs und Modellen. Dabei werden diese nicht direkt herausgegeben, sondern bei Aufrufen von Prozeduren wird die ID einer Datei als entsprechender Parameter übergeben. Die einzige direkte Möglichkeit Dateien hochzuladen existiert für das Hochladen von XES-Dateien in das System. Die Dateien für Modelle können nur erzeugt werden indem der Endpunkt der Process-Discovery (siehe Unterabschnitt 4.3.3) aufgerufen wird und dieser den Mining-Prozess startet, dessen Ergebnis wiederum eine Datei ist. Für die einzelnen Dateien existieren Einträge in der Datenbank. Dies dient nicht nur der besseren Verwaltung, sondern auch zusätzlichen Metainformationen, wie z. B. dem Algorithmus mit dem ein Modell erstellt wurde.

Abbildung 4.4.: Die Darstellung zeigt die Tabelle mit einer XES-Datei aus der Webapp. Über einen Popup-Dialog lässt sich der zu verwendende Algorithmus zum Mining auswählen.

Abbildung 4.5.: Liste der Petri-Netze aus der Webapp.

Datentyp: *Datei*		
Bezeichner:	Datentyp	Beschreibung
fileName	String	Der Name der Datei.
id	Integer	Identifikations-Zahl, welche für die jeweiligen Subtypen eindeutig sind.

Datentyp: *XES-Datei*		
Erweitert:	*Datei*	
Bezeichner:	Datentyp	Beschreibung
uploadDate	Date	Zeitpunkt, zu dem die XES-Datei hochgeladen oder erstellt wurde.

Datentyp: *Modell-Datei*		
Erweitert:	*Datei*	
Bezeichner:	Datentyp	Beschreibung
miningDate	Date	Zeitpunkt, zu dem das Modell durch Process-Discovery erstellt wurde.
miningAlgorithm	String	Name des Algorithmus, mit dem das Modell erstellt wurde
sourceOriginalName	String	Name der **XES-Datei**, aus der das Modell erstellt wurde.
sourceId	Integer	ID der **XES-Datei**, aus der das Modell erstellt wurde.

Datentyp: *PNML-Datei*	
Erweitert:	*Modell-Datei*

Datentyp: *BPMN-Datei*	
Erweitert:	*Modell-Datei*

Endpunkt: Liste der XES-Dateien	
Beschreibung:	Liefert eine Liste aller erstellten oder hochgeladenen XES-Dateien.
URL:	/xes
Type	GET
Ergebnis	
Ein JSON-Array mit Elementen vom Typ **XES-Datei**.	

Endpunkt: Hochladen einer XES-Datei		
Beschreibung:	Hochladen einer XES-Datei über „multipart/form-data“. (Nützlich um das System auch mit anderen Daten als die aus dem Elasticsearch zu verwenden und zu testen.)	
URL:	/xes/upload	
Type	POST	
Parameters		
Bezeichner:	Datentyp	Beschreibung
file	Datei	Datei, welche hochgeladen wird.

Endpunkt: Liste der PNML-Dateien	
Beschreibung:	Liefert eine Liste aller erstellten oder hochgeladener PNML-Dateien.
URL:	/pnml
Type	GET
Ergebnis	
Ein JSON-Array mit Elementen vom Typ **PNML-Datei**.	

Endpunkt: Liste der BPMN-Dateien	
Beschreibung:	Liefert eine Liste aller erstellter oder hochgeladener BPMN-Dateien.
URL:	/bpmn
Type	GET
Ergebnis	
Ein JSON-Array mit Elementen vom Typ **BPMN-Datei**.	

4.3.3. Endpunkte zur Process-Discovery

Folgende Endpunkte dienen dazu, den Prozess der Process-Discovery zu starten. Dabei werden dem Endpunkt die ID eines Logs und der entsprechende Algorithmus übergeben. Der Prozess wird asynchron gestartet, da das Mining sehr lange Laufzeiten besitzen kann. Nach der erfolgreichen Terminierung des Vorganges wird das erzeugte Modell in den entsprechenden Endpunkten angezeigt. Aufgrund von technischen Einschränkungen des ProM-Framework ist das Ergebnis des α-Algorithmus nur ein Petri-Netz, während beim evolutionären Algorithmus sowohl ein Petri-Netz als auch eine BPMN erzeugt wird.

Endpunkt: Liste der verfügbaren Algorithmen.	
Beschreibung:	Liste aller Algorithmen, welche für die Process-Discovery verwendet werden können.
URL:	/mining/algorithms
Type	GET
Ergebnis	
Ein JSON-Objekt as Wörterbuch mit dem Bezeichner des Algorithmus als Schlüssel und einer kurzen Beschreibung als Wert.	

Endpunkt: Starte Mining-Prozess.		
Beschreibung:	Stößt asynchron den Prozess zu Erstellung eines Prozessmodells aus einem Log an.	
URL:	/mining/run	
Type	POST	
Parameters		
Bezeichner:	Datentyp	Beschreibung
xesId	Integer	ID des zu verwendenden Logs (XES-Datei).
miningAlgorithm	String	Bezeichner des Mining-Algorithmus, der verwendet werden soll.
Ergebnis		
Bezeichner:	Datentyp	Beschreibung
status	String	Nachricht, welche das erfolgreiche Starten des Vorgangs ausgibt.

4.3.4. Endpunkte für die Graphen der Modelle

Die folgenden Endpunkte liefern die vorhandenen Modelle zurück. Dabei wandeln sie die in den Dateien vorliegende XML-basierte Darstellung in eine für die Webapp gut zu verarbeitende JSON-Darstellung um.

Datentyp: *Graph*		
Beschreibung:	Beschreibt die Elemente eines Graphen zur Darstellung eines Petri-Netzes oder eines BPMN.	
Bezeichner:	Datentyp	Beschreibung
nodes	Array	Array mit Elementen des Typ *Node*.
links	Array	Array mit Elementen des Typ *Link*.

Datentyp: *Node*		
Bezeichner:	Datentyp	Beschreibung
type	String	Beschreibt den Knotentyp des Elements. Falls der zugrunde liegende Graph ein Petri-Netz ist, sind mögliche Werte „transition“ und „place“. Bei BPMN sind „start“, „end“, „task“ und die Bezeichnungen für Gateways („AND“, „OR“, „XOR“) möglich.
id	Integer	Ein für den Graphen eindeutiger identifizierender Wert.
desc	String	(optional) Beschriftung des Knoten. Daher nur bei Knoten mit Beschriftung.
tau	Boolean	(optional) Beschreibt bei Transitionen, ob es sich um eine stille Transition handelt. Der Standard Wert, wenn nicht angegeben ist „false“.
start	Boolean	(optional) Beschreibt, ob es sich bei diesem Element um das eindeutige Startelement handelt. Wenn nicht angegeben: „false“.
end	Boolean	(optional) Beschreibt, ob es sich bei diesem Element um das eindeutige Endelement handelt. Wenn nicht angegeben: „false“.

Datentyp: *Link*		
Bezeichner:	Datentyp	Beschreibung
source	Integer	ID des *Node*-Objektes, aus dem die Kante entspringt.
target	Integer	ID des *Node*-Objektes, das Ziel der Kante ist.

Endpunkt: Graph eines PNML.		
Beschreibung:	Liefert den Graph einer PNML Datei.	
URL:	/graph/pnml	
Type	GET	
Parameters		
Bezeichner:	Datentyp	Beschreibung
id	Integer	Id der PNML-Datei.
Ergebnis		
Ein Objekt des Typ *Graph*, welcher das Petri-Netz darstellt.		

Endpunkt: Graph eines BPMN.		
Beschreibung:	Liefert den Graph einer BPMN Datei.	
URL:	/graph/bpmn	
Type	GET	
Parameters		
Bezeichner:	Datentyp	Beschreibung
id	Integer	Id der BPMN-Datei.
Ergebnis		
Ein Objekt des Typ *Graph*, welcher das BPMN darstellt.		

4.4. Erstellen der Traces

Der erste Verarbeitungsschritt ist die Erstellung der Extensible Event Stream Dateien aus den Einträgen, welche das untersuchte Anwendungssystem in Log Dateien schreiben und die dann über das Elasticsearch zugänglich gemacht werden. Das Format eines solchen Eintrages zeigt Listing 4.1 an einem Beispiel. Als aller erstes ist es nötig ein nicht die Einträge aller Aktionen zu betrachten, da sonst versucht werden würde ein Modell für das gesamte System zu erstellen. Dies wäre nicht nur sehr aufwendig, sondern auch das Ergebnis wäre so komplex, dass es nicht für einen Menschen direkt verständlich wäre und somit auch keinen Nutzen darstellen würde.

Als Filter wurde die Beteiligung eines Ablaufes an einem Service genutzt, da leider keine Zuordnung von tatsächlichen Use-Cases des Systems zu den einzelnen Abläufen gemacht werden konnte. Da für die meisten Use-Cases ein Service existiert der ausschließlich von diesem aufgerufen wird, kann über die Filterung nach diesem ein Log für einen Use-Case erstellt werden.

Zusätzliche Parameter sind noch der Zeitraum und die ungefähre Menge an zu erstellenden Traces. Damit werden mit dem Filter nach einem Service eine Menge von Tracking-IDs (siehe Unterabschnitt 4.1.3) erstellt.

Listing 4.1 Beispielhafte Darstellung eines Eintrages in Elasticsearch in JSON (NoSQL-Stil).

```
{
"APP" : "System1",
"Service" : "ServiceA",
"message" : "Application exit with error code 0.",
"timestamp" : "2015-10-21T16:29:00-08:00",
"trackingID" : "1234-ABCD-123456789ABC-DEF1-2345"
}
```

Workaround wegen Textindizierung der Tracking-ID

Da Elasticsearch auf Apache Lucene basiert, einem Framework für die Indizierung von Texten, ist die Standardindizierung der Volltextindizierung. Da die Tracking-ID aus mehreren Strings (Hexdarstellung) besteht, welche mit Bindestrich miteinander verbunden sind (siehe Listing 4.1), wird die Tracking-ID in wie ein Satz mit mehreren Worten angesehen. Dadurch sind nur die einzelnen Teile im Index, nicht aber die komplette ID, und es ist nicht möglich direkt eine Anfrage zu stellen, welche die mit einem Service assoziierten Tracking-IDs zurückliefert. Die beste Lösung wäre die korrekte Indizierung der Tracking-ID zu veranlassen. Da dies jedoch für Dauer der Arbeit eine lange dauernde Aufgabe wäre, die nur unnötige Verzögerung mit sich gebracht hätte, wurde folgender Workaround gewählt:
Es werden die mit einem Service assoziierten Teilwörter der Tracking-ID vom Elasticsearch gefordert. Aus diesen werden die Wörter der Länge zwölf herausgefiltert, da es in jeder ID nur einen Teil dieser Länge gibt. Und letztendlich ein Eintrag von Elasticsearch gefordert, der dieses Wort in der Tracking ID beinhaltet, um somit eine vollständige Tracking ID zu erhalten.

Für die gegebene Menge IDs werden nun alle Einträge angefordert, die entweder “application enter“ oder mit “application exit“ beinhalten. Logeinträge mit diesen beiden Texten werden bei Start und Beendigung eines Services erstellt. Die Sicht wird somit auf diese beiden Ereignisse beschränkt.

Der letzte Schritt ist die gegebenen Eigenschaften der Einträge auf definierte Attribute von XES zu übertragen. Dabei werden die Gesamten im Vorgang erstellen Daten als ein Log angesehen. Die Gesamtheit einer Tracking-ID zugehörigen Elemente sind ein Trace und die einzelnen Einträge, geordnet nach ihrer zeitlichen Abfolge, sind die Events. Einige Attribute der Events sind folgendermaßen gegeben:

$$
\begin{aligned}
\text{time:timestamp} &= \text{timestamp} \\
\text{concept:name} &= \text{APP:Service} \\
\text{lifecycle:transition} &= \begin{cases} \text{start} & \text{, falls der Eintrag “application enter“ beinhaltet} \\ \text{complete} & \text{, falls der Eintrag “application exit“ beinhaltet} \end{cases}
\end{aligned}
$$

Mit Hilfe von OpenXES, der Referenzimplementierung des XES-Standards, wird der beschriebenen Log erstellt und in das XML-Format serialisiert. Zusätzlich wird ein Eintrag in der SQL-Datenbank erstellt, welcher der besseren Verwaltung der Dateien dient.

4.5. Mining der Modelle

Der nächste Verarbeitungsschritt ist das Erstellen von Modellen für die gegebenen Logs. Dafür wird auf das ProM-Framework zurückgegriffen. Der Aufruf der Algorithmen wurde über das gegebene Command-Line-Interface [pro15] aufgeführt. Dieses bekommt als Eingabe die Datei eines BeanShell[bea] Skriptes gegeben, welches darauf im Kontext des ProM-Framework ausgeführt wird. Die Sprache ist mit einigen Befehlen für die Verwendung der einzelnen Komponenten angereichert. Dadurch werden die Schritte ausgeführt, die normalerweise über das GUI des ProM-Framework ausgeführt werden:

1. Laden des XES-Logs
2. Mining eines Modells aus dem Log
3. (optional: falls nötig Konvertieren des Ausgabemodells in die gewünschte Notation)
4. Serialisieren des Modells (Schreiben in Datei)

Es existieren einige Plugins des Framework, welche sich nur über das GUI aufrufen lassen und die nötigen Schnittstellen für die Kommandozeilen Ausführung nicht implementieren. Trotzdem ist diese Lösung besser, als die Komponenten des ProM-Frameworks direkt einzubinden. Listing 4.2 zeigt das Script für den α-Algorithmus. Für den Aufruf des evolutionären Algorithmus fehlt leider das Interface für die Verwendung ohne UI (in der verwendeten ProM-Version 6.5). Daher musste hier der weg über direkte Aufrufe gegangen werden, um die Funktionalität trotzdem nutzen zu können. Dies forderte an mehreren Stellen auch die Nutzung von Reflexion, um versteckte Methoden zugänglich zu machen. So wurde aber eine korrekte Implementierung der Algorithmen und damit die Qualität der erstellten Modelle sichergestellt. Für die Parameter des evolutionären Algorithmus wurden sinnvolle Werte (die vom ProM empfohlenen) gewählt und diese im Skript hart-codiert. Dies verringert die Komplexität des Systems und verhindert Verwirrung beim User die durch Eingabe der Werte entstehen könnten.

Listing 4.2 BeanShell-Skript, welches für die Ausführung des α-Algorithmus aus dem ProM verwendet wurde.

```
Properties prop = System.getProperties();

log = open_xes_log_file(
      prop.getProperty("de.fabiangajek.prom.cli.xesFile"));

net_and_marking = alpha_miner(log);

net = net_and_marking[0];

File net_file = new File(
      prop.getProperty("de.fabiangajek.prom.cli.pnmlFile"));

pnml_export_petri_net_(net, net_file);

System.exit(0);
```

4.6. Besonderheiten der Hierarchisierung

Die in Abschnitt 3.4 besprochenen Konzepte zum Ausnutzen von Hierarchien wurden auch implementiert und werden in diesem Abschnitt vorgestellt. Strukturiert ist dies nach den Verarbeitungsschritten.

4.6.1. Hierarchie in der Erstellung der Traces

Für die Erstellung der Traces war von Bedeutung, dass in den Logs nicht dokumentiert wurde, von welchem Service ein andere Service aufgerufen wurde. Nach Rücksprache mit einem Experten des Systemes und durch empirische Beobachtung der Logdaten wurde davon ausgegangen, dass es innerhalb einer Ausführung eines Prozesses keine parallelen Abläufe gibt. Daher werden die betrachteten Einträge, die Start und Ende von Prozessen darstellen, einfach wie bei der Unterprogramm Ausführung mit einem Stack verwaltet. Die Einträge liegen in zeitlich sortierter Form vor, es existiert ein zu Anfang leerer Stack und in die darauf wird Folgendes für jeden Eintrag eines Traces ausgeführt:

1. Falls der Eintrag einen Start dokumentiert, lege den Bezeichner auf den Stack.
2. Nutze den vollständigen Bezeichner, der durch den Stack beschrieben ist, für den Eventeintrag in die XES-Datei. Die Notation wird wie in Unterabschnitt 3.4.1 beschrieben umgesetzt.

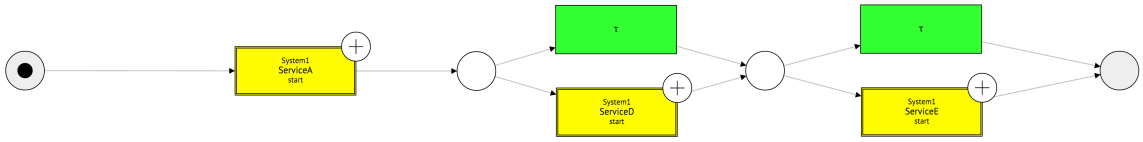


Abbildung 4.6.: Die Abbildung zeigt Darstellung eines Petri-Netzes, wie sie im Prototyp verwendet wird.

3. Setzte den Lebenszyklus des Events auf den entsprechenden Wert (“start“ oder “complete“).
4. Falls der Eintrag ein Ende dokumentiert, entferne das oberste Element vom Stack.

Die so erstellten Logs können ohne Änderungen von einem Mining-Algorithmus verwendet werden.

4.7. Visualisierung von Petri-Netzen und BPMN

Die Darstellung der Graphen erfolgt innerhalb der Webapp. Dazu wird, um eine gut skalierbare Darstellung zu bekommen, ein SVG-Element verwendet und dessen Elemente mithilfe der D3-Bibliothek manipuliert. Die Visualisierung von Petri-Netzen und BPMN unterscheidet sich dabei wenig, wenn Transitionen analog zu Task angesehen werden.

Für die Darstellung der Modelle als Graphen soll das D3-Framework und die damit gegebene Implementierung von kräftebasierten Layouts (siehe Unterabschnitt 2.7.1) verwendet werden. Da D3 in Javascript implementiert und auf die Verwendung in einem Browser ausgelegt ist, sollte die Komponente der Visualisierung am Besten als Webapp ausgelegt sein und über Schnittstellen mit dem Rest des Systems kommunizieren. Zur besseren Verarbeitung der Daten in Javascript und D3 ist es außerdem sinnvoll die übergebenen Daten schon im JSON-Format zu übergeben.

Um die Elemente der Graphen übersichtlich zu verteilen, wird ein kräftebasiertes Layout verwendet (siehe Unterabschnitt 2.7.1). Da es möglich ist die Position einzelnen Knoten (Stellen, Transitionen, Task, ...) im kräftebasierten Layout einzufrieren, wird dies initial für den Start- und Endknoten getan. Außerdem wird der Startknoten links und der Endknoten rechts im Bereich des Layouts angeordnet, damit sich der Graph von links nach rechts ausrichtet. Dies dient der besseren Erkennung des Modells, da durch die geordnete Darstellung übersichtlicher ist und sich immer horizontal ähnlich ausrichtet. Abbildung 4.6 und Abbildung 4.7 zeigen wie das für ein globales Modell (alle Unterprozesse sind eingeklappt) aussieht.

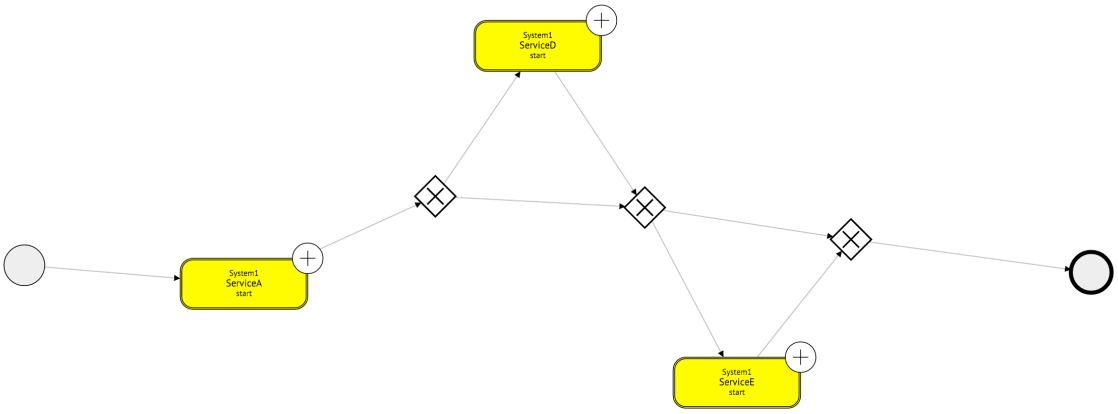


Abbildung 4.7.: Die Abbildung zeigt Darstellung eines BPMN, wie sie im Prototyp verwendet wird.

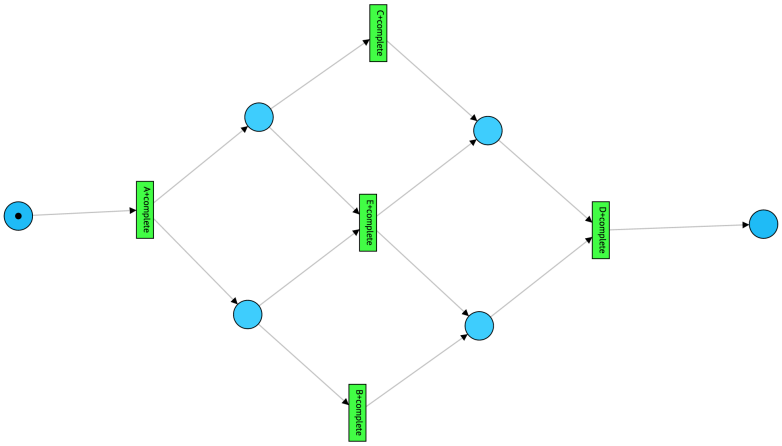


Abbildung 4.8.: Visualisierung eines Petri-Netzes wie sie in einer ersten Version des Systems verwendet wurde.

4.7.1. Interaktion mit der Visualisierung des Graphen

Um den Graphen besser darzustellen, sind dem User einige Interaktionsmöglichkeiten gegeben, welche die Darstellung beeinflussen:

Einfrieren eines Knoten Über ein Kontextmenü ist es möglich einzelne Task, Transitionen oder anderen Elemente einzufrieren. Dadurch werden die Positionen vom kräftebasierten Layout nicht verändert. Zusammen mit dem **Verschieben eines Knoten** kann so die Darstellung manuell noch weiter optimiert werden. Wenn die Position eines

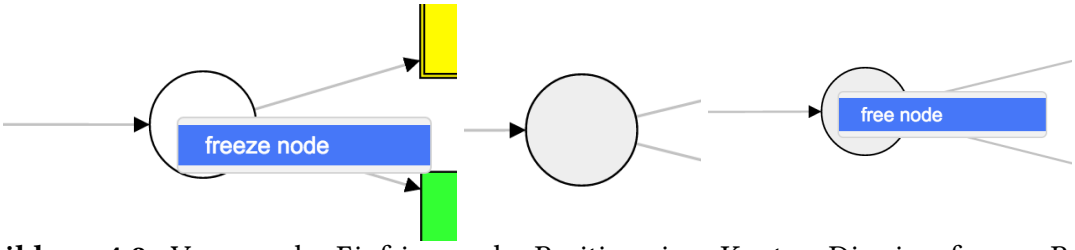


Abbildung 4.9.: Vorgang des Einfrierens der Position eines Knoten. Die eingefrorene Position wird über eine dunklere Farbe dargestellt.

Knoten eingefroren ist, wird dies visuell über eine leicht dunklere Farbe des Elementes dargestellt. Über das Kontextmenü ist auch das Aufheben des Einfrieren möglich.

Verschieben eines Knoten Durch das Drücken, Halten der linken Maustaste und dem Verschieben der Maus, kann ein User beliebige Knoten an eine andere Stelle schieben. Während und nach dem Verschieben wird das Layout Kräfte basiert optimiert. Dabei ist das gezogene Element während des Vorganges auf die Position der Maus fest. Nach dem Loslassen der Maustaste bleibt der Knoten, abhängig davon, ob die Position eingefroren wurde, stehen oder seine Position wird durch die Kräfte optimiert.

Zoomen und Verschieben Über das Mausrad und durch Ziehen mit gedrückter linker Maustaste (außer auf Knoten) lässt sich die Darstellung des Graphen fast beliebig verschieben, vergrößern und verkleinern. Das Zoomen der Darstellung wird dabei um die Position des Mauszeigers vorgenommen.

4.8. Hierarchie in der Visualisierung der Modelle

Dadurch, dass die globalen Bezeichner der einzelnen Task bis in das Modell mitgezogen werden, ist es möglich dies auszunutzen, um Subprozess aus- und einzublenden. Initial werden alle Submodelle ausgeblendet, um einen möglichst kleinen Graphen zu bekommen, dessen visuelle Optimierung damit vereinfacht wird. Visuell werden Transitionen oder Tasks, deren Subprozess ausgeblendet wird, gelb hinterlegt und ein Plus ist an der oberen rechten Ecke zu sehen, über das sich der Subprozess einblenden lässt (siehe Abbildung 4.10). Das Anzeigen des Subgraphen läuft ab wie beim Zusammensetzen der lokalen Modelle (siehe Unterabschnitt 3.4.3). Expandierte Transitionen sind grün hinterlegt und besitzen ein Minus in der rechten oberen Ecke, dieses dient zum erneuten Ausblenden des Unterprozesses. Transitionen ohne Subprozess sind auch grün hinterlegt, ihnen fehlt aber das Minus zum Einklappen.

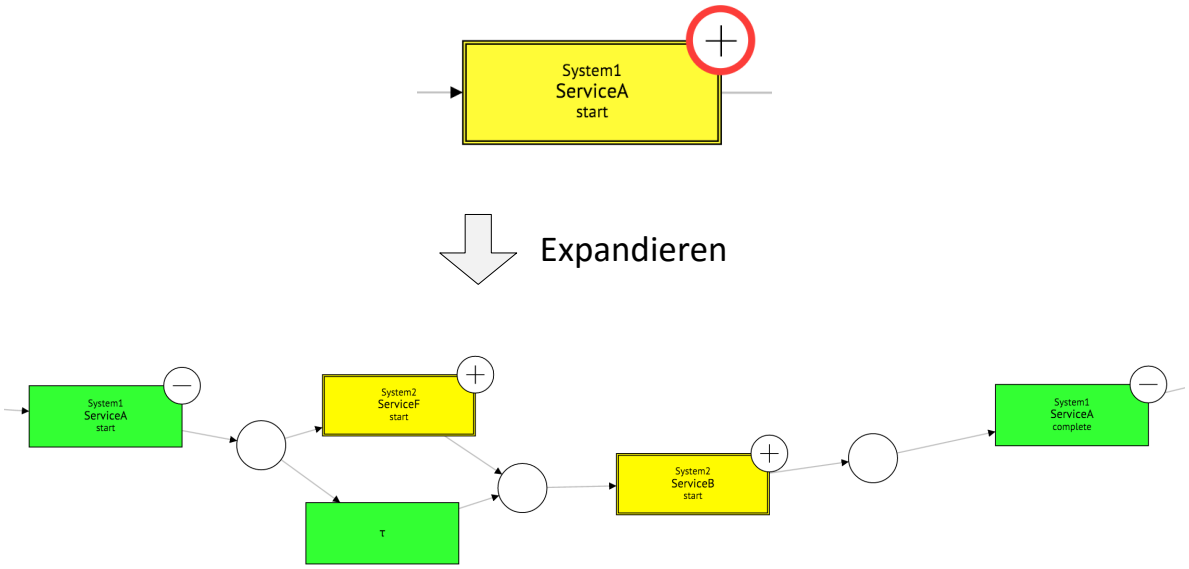


Abbildung 4.10.: Durch einen Klick auf das Plus eines Knoten wird dieser expandiert. D. h. die Transition wird ersetzt durch den entsprechenden Subprozess (siehe Abbildung 3.3).

4.9. Anzeigen des zugrundeliegenden Logs

Um nicht nur das Modell zu verstehen, sondern auch warum es so aussieht, werden zum Graphen auch die einzelnen Traces angezeigt, aus denen dieser entstanden ist.

Dabei werden, wie in Abbildung 4.11 zu sehen ist, die einzelnen unterschiedlichen Traces angezeigt. Wenn mehrere äquivalente Traces im Log vorhanden sind, so ist dies über die Anzahl (engl. Count) angegeben. Traces mit einer hohen Anzahl stellen das häufige Verhalten dar. Traces mit kleineren Anzahlen sind auch interessant, da diese entweder Randverhalten oder möglicherweise auch Fehlverhalten darstellen.

Um noch genauer zu untersuchen welcher Teil des Modells durch welche Traces entstanden ist, lässt sich der Log nach einzelnen Transitionen filtern.

4.10. Visualisierung von Traces in einem Model

Um die Plausibilität des Modells zu verbessern, ist es möglich, einen oder mehrere Traces visuell im Modell anzuzeigen. Dazu wurden zwei verschiedene Ansätze implementiert. Der Erste verbindet nur alle Transitionen aus einem Trace in genau der Reihenfolge, in der sie

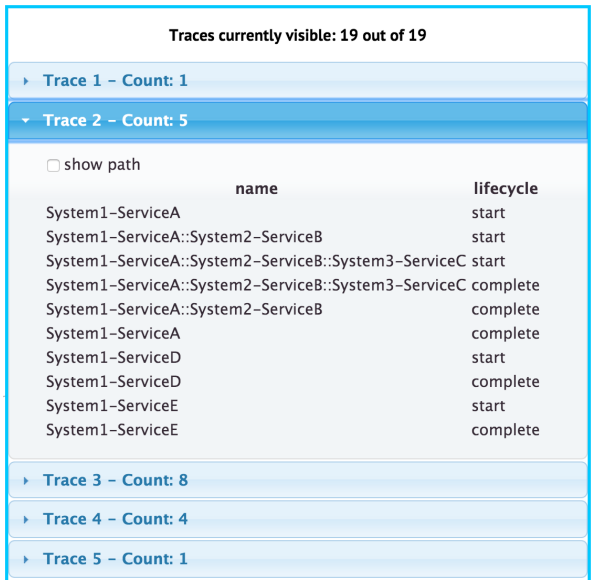


Abbildung 4.11.: Die einzelnen Traces des Logs werden mit dem Modell angezeigt, um Verständnis und Plausibilität des Modells zu unterstützen.

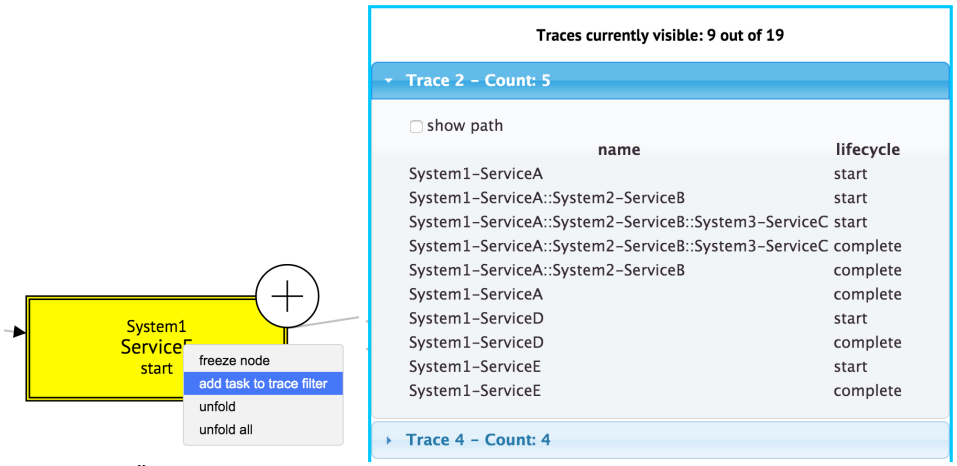


Abbildung 4.12.: Über das Kontextmenü lassen sich die Traces nach Transitionen filtern.

auch dort vorkommen (ausgeblendete Transitionen werden immer übersprungen). Dies ist in Abbildung 4.13 dargestellt für den Trace 2 der in Abbildung 4.11 zu sehen ist.

Der zweite Ansatz versucht die Bewegung der Marken zu visualisieren, indem es eine Linie durch die Pfade zeichnet, an denen sich eine Marke für den Vorgang bewegen würde. Dies ist in Abbildung 4.14 abgebildet.

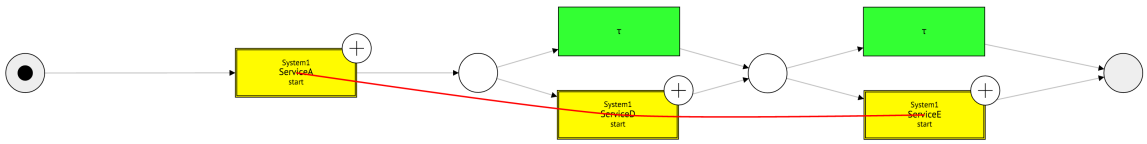


Abbildung 4.13.: Visuelle Darstellung eines Traces durch Verbindung der entsprechenden Transitionen.

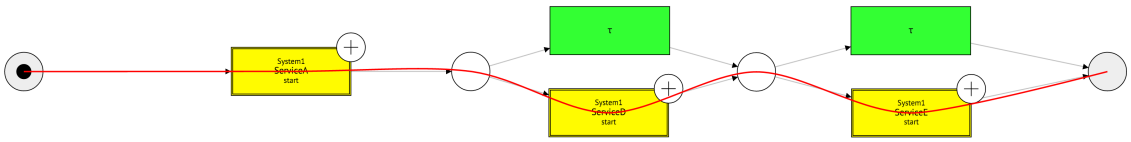


Abbildung 4.14.: Visuelle Darstellung eines Traces durch die Markierung des Pfades der beteiligten Marken.

5. Evaluation durch Expertenstudie

Dieses Kapitel beschreibt die Evaluation des implementierten Systems mittels einer Expertenstudie. Dazu wurden einer kleinen Runde von Experten des untersuchten Anwendungssystems das Tool vorgestellt und anschließend von diesen evaluiert. Der erste Abschnitt beschreibt die Durchführung der Studie, während der zweite Teil die gewonnenen Erkenntnisse aufbereitet.

5.1. Durchführung der Expertenstudie

Die Expertenstudie wurde als Kombination aus Expertenbefragung und Fokusgruppe durchgeführt. Das bedeutet, alle Probanden waren gleichzeitig in einem Raum. Dabei wurde ihnen der Prototyp vorgestellt und sie mussten dazu Fragen beantworten. Außerdem sollten sie in einer anschließenden Diskussion noch weitere Erkenntnisse ausarbeiten. Die bei der Studie verwendeten Dokumente befinden sich in Abschnitt A.1. Zur Unterstürzung der Demonstration wurde den Probanden eine Anleitung ausgehändigt, welche die vorgestellten Interaktions- und Analysemöglichkeiten noch einmal erklärt. Bei der Durchführung der Studie wurde ein Modell verwendet, das aus den Logdaten des untersuchten Anwendungssystemes generiert wurde. Den Experten wurde zum Vergleich ein existierendes Modell des gleichen Ablaufes gezeigt. Das generierte Modell deckte aufgrund von mangelnder Abdeckung der verwendeten Logdaten nur einen Teil der Abläufe ab. Daher war der Vergleich der Modelle leicht eingeschränkt.

Für die Studie haben sich vier Experten zur Teilnahme bereit erklärt. Die Teilnehmer waren 31 bis 50 Jahre alt und die jeweilige Berufserfahrung reichte von sieben bis 25 Jahre. Dabei haben zwei der vier Experten promoviert. Die Aufgaben der Teilnehmer im Unternehmen sind thematisch beim Betrieb von Systemen und im Management angeordnet.

Bei der Durchführung sollte nach Plan die Diskussion nach der Beantwortung der Fragen sein. Durch Zwischenfragen während der Demonstration des Programmes wurde die Diskussion praktisch in die Demonstration vorgezogen. Die entstandenen Erkenntnisse sind trotzdem sehr aufschlussreich und gut zu verwerten.

5.2. Ergebnisse der Expertenbefragung

In diesem Abschnitt werden die Ergebnisse der Expertenstudie besprochen. Dabei werden zum einen die Antworten der Probanden zu den Multiple-Choice Fragen analysiert (siehe Abschnitt A.2 - Antworten der Multiple-Choice Fragen), zum anderen die weiteren Anmerkungen der Probanden zusammengefasst.

5.2.1. Analyse der Multiple-Choice-Fragen

Durch die geringe Anzahl an Probanden sind die Aussagen der Antworten eingeschränkt. Trotzdem lassen sich Fragen nutzen, bei denen die Antworten einen klaren Trend zeigen.

Die Multiple-Choice-Fragen hatten eine Skala von 1 bis 5. Dabei sollte jeweils eine Aussage zum Prototypen beantwortet werden. Die Fragen waren zu den folgenden Themen:

1. Bedarf für Process-Mining im Unternehmen.
2. Nutzen der verwendeten Visualisierungen.
3. Nutzen von Process-Mining im Allgemeinen.

Die Antworten zeigen die Meinung der Experten, dass eine solche Software zum Mining von Prozessen in ihren (und auch in anderen) Unternehmen Verwendung finden würde. Die unterschiedlichen Antworten bei den Fragen nach Häufigkeit der Verwendung und persönlicher Unterstützung lassen sich vor allem auf unterschiedliche Aufgabengebiete der Experten zurückführen. Daraus folgt auch eine andere Sicht auf die Anwendbarkeit der Software. Nutzen sahen die Experten vor allem in der Erkennung von Abweichung, gerade auch im Zusammenhang mit dem Finden von Fehlern und bei der Prozessdokumentation.

Für die unterstützenden Elemente lässt sich sicher sagen, dass die textuelle Auflistung der Traces unübersichtlich ist. Dies mag auch an komplexen global Bezeichner liegen oder einfach an der Unübersichtlichkeit von vielen gleichen Zeichen und Wörtern. Die visuelle Unterstützung dagegen kam besser an und die Meinung der Experten ist es, dass die Visualisierung des Kontrollflusses der einzelnen Traces “sehr stark“ das Verständnis des Modells fördert. Die Darstellung des Graphen mit einem interaktiven kräftebasierten Layout wurde auch gut angenommen.

Nach der Meinung der Probanden sind Teile der Funktionalität eines solchen Programmes gegeben. Das System nutzt auch schon vorhandene Systeme, welche die Fehlererkennung unterstützen sollen (Elasticsearch, Kibana). Da keine der vorhandenen Systeme Modelle aus den Daten erstellt, sind große Teile aber auch noch nicht abgedeckt.

5.2.2. Analyse der Diskussion und Freitextfragen

Die Probanden wurden zur Diskussion angeregt und sie bekamen Freitextfragen. In diesen sollte sie beurteilen wie ein dem Prototypen entsprechendes Programm verwendet werden kann, was ihnen gefallen hat oder was sie verbessern würden.

Die Reaktionen über das vorgestellte Programm reichen von vielen positiven Rückmeldungen über das interaktive kräftebasierte Layout des Graphen bis zu einigen negativen Anmerkungen zur Unvollständigkeit des Modells.

Die Meinung der Probanden zu den Anwendungsmöglichkeiten entsprechen sehr den Anwendungen des Process-Mining im Allgemeinen: “Verifikation von Prozessen“, “Analyse“, “Optimierung [der Prozesse]“, “Reverse Engineering“ der Prozesse. Dies zeigt auch das Bedürfnis von Systemen mit solcher Funktionalität, denn das Process-Mining sieht andere Algorithmen und Prozesse als die implementieren als Lösung für einige dieser Anwendungen vor.

Der wichtigste Kritikpunkt des Systems war die Unvollständigkeit der vorhandenen Traces und deswegen auch des Modells. Dies zu lösen fordert einige Voraussetzungen. Denn um eine gute Process-Discovery, oder Process-Mining im Allgemeinen, durchführen zu können, müssen die Prozesse erst gut formal im Log dokumentiert werden [VDAAM+12]. Deshalb gehört zu einer reifen Implementierung eines Process-Mining-Systems auch eine informative und formale Dokumentation abgelaufener Prozesse. Durch die prototypischen Eigenschaften des implementierten Programmes und einem nicht auf das Process-Mining ausgelegten Systemes waren solche Voraussetzungen nicht gegeben.

Ein Kritikpunkt eines Experten war vor allem die Einschränkung der Prozesssicht auf eine Top-Level-Sicht. Es lassen sich zwar Unterprozesse einblenden, der erste Eindruck liegt aber auf dem Ablauf des Service, welcher beim Eintritt in das System aufgerufen wird. Unterprozesse, bei denen die entscheidenden Abläufe stattfinden, können in den Hintergrund rücken.

Als eine Verbesserung wurde vorgeschlagen, auch Veränderungen am Modell zu visualisieren. Wenn ein solches Tool produktiv in einem System verwendet würde, so ist es möglich, täglich Änderungen am Modell vorzunehmen. Der Vorschlag ist die Änderungen am Modell von einem Zeitpunkt zum Nächsten hervorzuheben. Dadurch könnten dann Prozessänderungen dargestellt und analysiert werden. Außerdem besteht die Möglichkeit, plötzlich auftretendes Fehlverhalten zu erkennen.

Es wurde gezeigt, dass Process-Mining-Tools viele Anwendungen finden und dringend benötigt wird. Trotz der prototypischen Implementierung waren die Experten schon an der Verwendung der Software interessiert. Jedoch wären für einen produktiven Einsatz noch Verbesserungen vorzunehmen, gerade auch in der Erstellung der Logs. Und außerdem existieren eine Vielzahl an Möglichkeiten zur Maximierung des Analysepotenzials eines Process-Mining-Tools.

6. Fazit und Ausblick

In diesem Kapitel werden zunächst das Konzept, die Implementierung und die Expertenstudie zusammengefasst. Darauf folgt ein Ausblick über mögliche Erweiterungen am Konzept und an der Implementierung des Prototypen.

Fazit

In dieser Arbeit wurden die Anwendbarkeit der Process-Discovery auf Anwendungssysteme und ein Ansatz zur Optimierung der Darstellung und dem Verständnis des erstellten Kontrollflussmodells untersucht.

Das Konzept des Verarbeitungsprozesses wurde in drei Schritte aufgeteilt: zuerst die Vorverarbeitung der Logdaten des Anwendungssystems, dessen Aufgabe die Filterung auf relevante Einträge und die Umwandlung in ein standardisiertes Log-Format für Process-Mining ist. Folgenden wird ein Prozessmodell aus den Daten erstellt mit Hilfe eines Process-Discovery-Algorithmus. Schlussendlich werden Modell und Logdaten verwendet, um dem Nutzer eine visuelle Analyse zu bieten, welche nicht nur eine Darstellung des Modells ist, sondern auch eine Verbindung von aufgenommen Abläufen und Prozessmodell erstellt.

Der implementierte Prototyp wurde als eine Webapp mit einem Backend implementiert. Die Anwendung auf dem Server bietet dabei Funktionalität zum Erstellen von Logs aus einer Elasticsearch Instanz und zum Mining von Petri-Netzen und BPMN. Die Webapp stellt die Modelle grafisch dar und nutzt dabei ein kräftebasiertes Layout. Zusätzlich werden die zugrunde liegende Abläufe textuell aufgelistet und auch visuell in der Darstellung angezeigt.

Für die Evaluation des Ansatzes wurden Daten eines Anwendungssystems eines Automobilherstellers verwendet. Das System wurde danach mittels einer Expertenstudie analysiert, deren Teilnehmer Mitarbeiter eben dieses Automobilherstellers waren. Die Ergebnisse der Studie zeigten auf, dass Process-Discovery in Anwendungssystemen großes Potenzial hat und das Bedürfnis zur Analyse der abgelaufenen Prozesse ebenfalls sehr groß ist. Auch die visuelle Unterstützung wurde gut bewertet und half dem Verständnis der Modelle.

Ausblick

Das Konzept und der implementierte Prototyp zeigten sich in der Expertenstudie schon als sehr nützlich. Trotzdem existieren noch einige Aspekte, die einer Verbesserung unterzogen werden sollten oder deren Umsetzung den Nutzen sehr steigern könnten.

Optimierung der Hierarchisierung in der Visualisierung

Um das Erstellen der Modelle zu beherrschen, wurden Aufrufhierarchien eingeführt. Diese werden weiter in der Darstellung der Modelle verwendet um Unterprozesse aus- und einzublenden. Dadurch wird die Sicht sehr auf die Oberste Ebene fixiert und Unterprozesse sind meist weniger übersichtlich zu analysieren. Dies könnte verbessert werden, durch visuelle Darstellung der Aufrufebenen oder durch die Möglichkeit Unterprozesse als eigene Prozessmodelle in einem neuen Fenster darstellen zu lassen.

Verlauf von Änderungen

Eine mögliche Verbesserung, welche die Analyse sehr unterstützen würde, ist die Nutzung von regelmäßigen Erstellungen des Prozessmodelles um Veränderungen zu erkennen. Ein solches System zum Erstellen der Prozessmodelle macht es einfach jeden Tag ein Modell auf den aktuellen Logdaten zu erstellen. Dies gibt die Möglichkeiten Änderungen der Abläufe, gerade beim Umzug auf eine neue Version, durch das System zu erkennen, zu visualisieren und durch Experten zu analysieren.

Integration weiterer Algorithmen und Techniken des Process-Mining

Das Konzept und der implementierte Prototyp verwenden nur zwei Algorithmen der Process-Discovery. Daher sollte das System um weitere Algorithmen zum Erstellen von Prozessmodellen erweiterte werden, um die Ergebnisse vergleichen zu können. Einen größeren Nutzen hätte aber die Implementierung weiter Process-Mining Techniken. Sinnvoll wäre die Analyse von Laufzeiten einzelner Services um Abläufe zu optimieren. Oder eine weitere gute Erweiterung wäre die Überprüfung von Abläufen in Echtzeit auf die Konformität mit dem Modell. Schlussendlich wäre es sinnvoll das System um die Vielzahl an schon untersuchten Techniken des Process-Mining zu erweitern, um den Erkenntnisgehalt zu maximieren.

A. Anhang: Expertenstudie

A.1. Dokumente der Expertenbefragung

Anmerkung: Die Darstellungen des Programmes wurden aus Datenschutzgründen zensiert. Daher entsprechen die Abbildungen nicht die den Experten vorgelegten.

Einverständniserklärung zur Expertenstudie

Es wird eine Expertenstudie zu einer Bachelorarbeit der Universität Stuttgart durchgeführt. Die Studie ist wie folgt aufgebaut.

1. Darstellung des Ablaufes der Studie und Belehrung zu den Rechten
2. Allgemeine Fragen zur Person der Probanden
3. Darstellung der allgemeinen Bedienung des Programmes
4. Vorstellung eines möglichen Anwendungsfalles
5. Beantwortung der Testfragen
6. Fragen und Diskussion zum Programm

Hiermit werden Sie darüber aufgeklärt, dass Sie jederzeit die Möglichkeit haben

- die Studie abzubrechen, sowohl während der Einführung als auch jederzeit während der Durchführung,
- den Raum zu verlassen oder
- eine Pause zu machen.
- Außerdem können Sie Fragen die Sie nicht beantworten können/wollen (gerade bei den Fragen zur Person) einfach unbeantwortet lassen.

Die angegebenen Daten werden anonymisiert, vertraulich behandelt und nur zur Auswertung der Studie genutzt.

- Die Studie wird mindestens 30 Minuten dauern.

Mit meiner Unterschrift bestätige ich, dass ich die obigen Punkte gelesen habe und ihnen zustimme.

Ort, Datum _____

Unterschrift _____

Fragen zur Person

Berufsausbildung/Abschluss: _____

Aktuelle Aufgaben im Unternehmen: _____

_____

_____

Berufserfahrung (ca.): _____ Jahre

Geschlecht: [ ] m [ ] w

Alter _____

Fragen zum Programm

Gehen Sie im Folgenden davon aus, das Programm sei vollständig implementiert und marktreif.

Wie schätzen Sie den Bedarf eines solchen Programmes im Unternehmen ein?
Von 1 (kein Bedarf) bis 5 (sehr hoher Bedarf)

1	2	3	4	5

Wenn ein solches Programm in ihrem Unternehmen genutzt würde, wie häufig, denken Sie, würde es verwendet werden?

nie	Einmal im Monat	Einmal pro Woche	Täglich	Mehrmals täglich

Wie sehr würde ein solches Programm Sie bei ihren jetzigen Tätigkeiten unterstützen?
Von 1 (keine Unterstützung) bis 5 (starke Unterstützung)

1	2	3	4	5

Wo/wofür sehen Sie Einsatzmöglichkeiten für dieses Programm?

_____

_____

_____

_____

Wie beurteilen Sie die Plausibilität der Prozessmodelle?
Bedenken Sie dabei, dass nicht alle Aufrufe von Services geloggt werden.
Von 1 (gar nicht plausibel) bis 5 (komplett plausibel)

1	2	3	4	5

Wie sehr unterstützt die **textuelle** Auflistung der Traces die Plausibilität?
Von 1 (gar nicht) bis 5 (sehr stark)

1	2	3	4	5

Wie sehr unterstützt die **visuelle** Darstellung der Traces der Plausibilität?
Von 1 (gar nicht) bis 5 (sehr stark)

1	2	3	4	5

Wie sehr hilft die **visuelle** Darstellung der Traces dem Verständnis des Modells?
Von 1 (gar nicht) bis 5 (sehr stark)

1	2	3	4	5

Wie übersichtlich finden Sie die Darstellung der Graphen?
Von 1 (komplett unübersichtlich) bis 5 (sehr übersichtlich)

1	2	3	4	5

Wie ansprechend finden Sie die interaktive Darstellung der kräftebasierten Layouts (Bewegung der Elemente in Echtzeit)?
Von 1 (sehr störend) bis 5 (sehr ansprechend)

1	2	3	4	5

Denken Sie, dass bestehende Systeme die Funktionalität dieses Programms schon abdecken?
Von 1 (keine Funktionalität ist bereits abgedeckt)
bis 5 (die Funktionalität ist vollständig abgedeckt)

1	2	3	4	5

Wie schätzen Sie den Nutzen des Programmes ein, um Prozessabweichungen zu erkennen?
Von 1 (kein Nutzen) bis 5 (sehr großer Nutzen)

1	2	3	4	5

Wie schätzen Sie den Nutzen des Programmes zur Dokumentation von Prozessen ein?
Von 1 (kein Nutzen) bis 5 (sehr großer Nutzen)

1	2	3	4	5

Wie schätzen Sie den Nutzen des Programmes für die agile Entwicklung von Web Services?
Von 1 (kein Nutzen) bis 5 (sehr großer Nutzen)

1	2	3	4	5

Was hat Ihnen gut gefallen?

_____

_____

Was hat Ihnen nicht gefallen?

_____

_____

Was würden Sie verbessern?

_____

_____

Sonstiges / weitere Anregungen:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

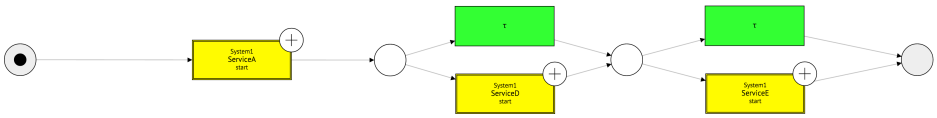
_____

_____

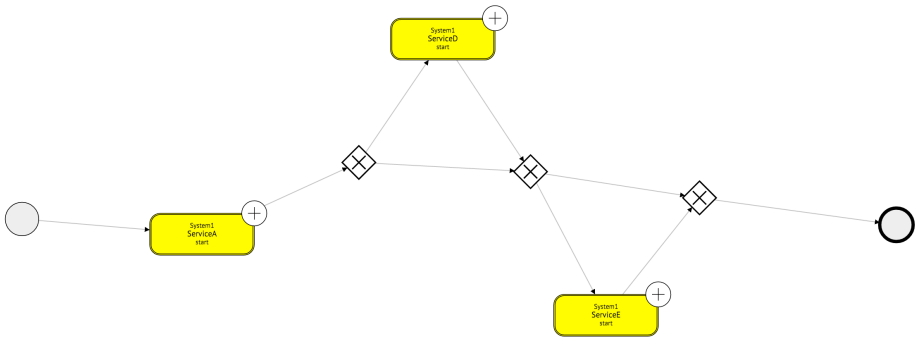
Vielen Dank für Ihre Teilnahme!

Anleitung Process-Mining-Tool

Visualisierung eines Petri-Netzes



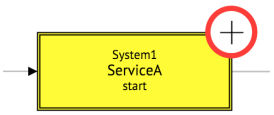
Visualisierung eines BPMN-Diagrammes

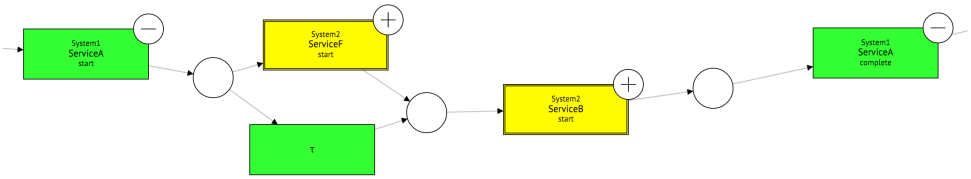


Die Interaktionsmöglichkeiten sind für beide Darstellungen gleich, daher werden in den Beispielen nur Petri-Netze verwendet.

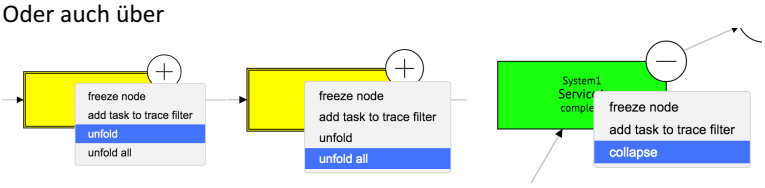
Expandieren von Subgraphen

Über die Plus und Minus Zeichen lassen sich Subgraphen expandieren und ausblenden.

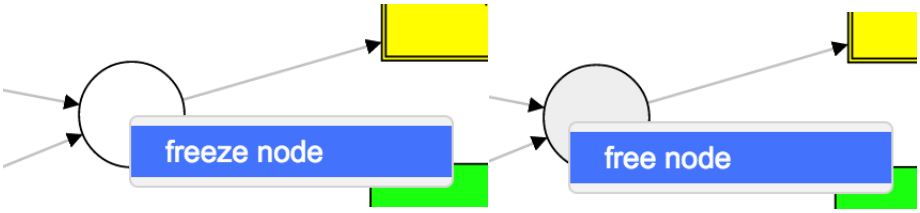




Oder auch über

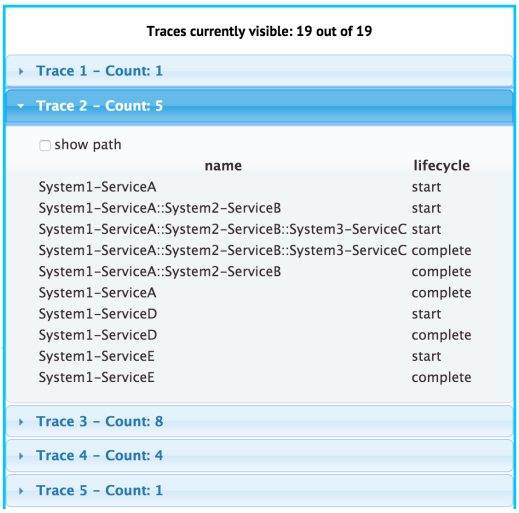


Einfrieren von Knoten

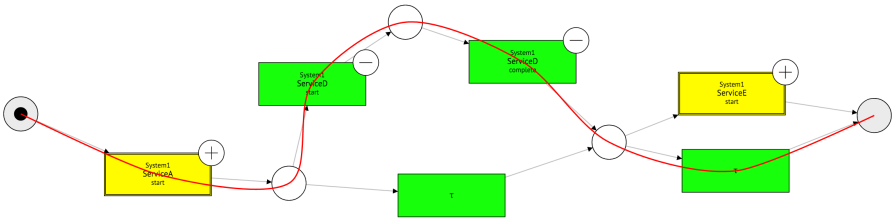


Darstellung der Traces

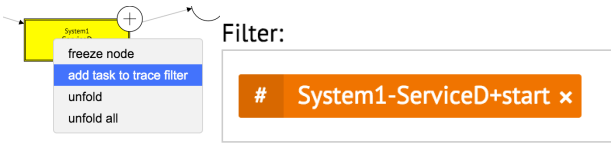
Rechts unter dem Punkt Traces sind im die geloggten Traces zu sehen. Äquivalente Abläufe sind zu einem Trace zusammengefasst und die Menge angegeben.



Über die „show path“-Checkbox können die Traces wiederum im Graphen angezeigt werden.



Um herauszufinden welche Abläufe zu einem Teil des Modells führen, lassen sich die Traces auch nach einzelnen Transitionen/Tasks filtern.



Ablauf Expertenbefragung

1. Begrüßung
2. Erklärung des Ablaufes
    2.1. Einführung und Belehrung
    2.2. Allgemeine Fragen zur Person der Probanden
    2.3. Vorstellung des Programmes – allgemeine Bedienung
    2.4. Darstellung eines Anwendungsfalles
    2.5. Ausfüllen des Fragebogens
    2.6. Allgemeine Frage und Diskussionsrunde
3. Austeilen der Einverständniserklärung
    3.1. Hinweis auf Recht jederzeit Aufzuhören oder den Raum zu verlassen.
    3.2. Einsammeln der Einverständniserklärung
4. Austeilend der Fragen zur Person
    4.1. Hinweis, dass die Angaben freiwillig sind
    4.2. Einsammeln der Fragen
5. Austeilen der Anleitungen
    5.1. Diese dienen zur Unterstützung der Demonstration
6. Demonstration
    6.1. Darstellung der Startseite
    6.2. Vorstellen der Datei Listen

6.3. Reiter zum Erstellen von XES-Traces mit Logs aus dem Elasticsearch

6.4. Öffnen eines Petri-Netzes und eines BPMN

6.4.1. Da alle Interaktionen bei der BPMN Visualisierung analog zu Petri-Netzen sind, werden im weiteren nur Petri-Netze zur Demonstration verwendet werden.

6.4.2. Am vorhanden Beispiel kurz Petri-Netze wiederholen

6.5.Vorstellen der Hierarchie des Graphen

6.5.1. Ausklappen eines Knoten

6.5.2. Einklappen eines Knoten

6.5.3. Ausklappen eines Knoten und all seiner Unterknoten

6.6. Vorstellung des Kräfte-basierten Layouts

6.6.1. Ziehen der Enden

6.6.2. Verschieben von Knoten, sodass sich ein anderes Layout bildet

6.6.3. Einfrieren von Knoten

6.6.3.1. Positionieren der eingefrorenen Knoten

6.7. Darstellung der Traces

6.7.1. Erklärung der Notation

6.7.2. Zwei Arten der Visualisierung der Traces im Modell

6.7.3. Filtern der Traces nach Transitionen / Anteil einer Verzweigung an Abläufen

7. Vergleich des vorgestellten Modells mit dem definierten Modell
    7.1. Darstellen des erstellen Modells im vorhanden Modell
    7.2. Hinweis auf die Unvollständigkeit der Log-Daten
        7.2.1. Daher auch nur Ausschnitt des Prozesses im erstellten Modell
8. Austeilen des Fragebogens
    8.1. 5-7 Minuten Zeit zur Beantwortung
    8.2. Warten bis alle fertig
    8.3. Hinweis, dass bei Sonstiges Gedanken eingetragen werden können die bei der Diskussion einfallen
9. Fragen und Diskussion
10. Abschluss und Danksagung

A.2. Antworten der Multiple-Choice Fragen

Frage 1

Frage Wie schätzen Sie den Bedarf eines solchen Programmes im Unternehmen ein?
Von 1 (kein Bedarf) bis 5 (sehr hoher Bedarf)

Antworten

5	5	5	4

Mittelwert $4,75$

Standardabweichung $0,43$

Frage 2

Frage Wenn ein solches Programm in ihrem Unternehmen genutzt würde, wie häufig, denken Sie, würde es verwendet werden?

Antworten

Täglich	Einmal im Monat	Täglich	Einmal im Monat

Frage 3

Frage Wie sehr würde ein solches Programm Sie bei ihren jetzigen Tätigkeiten unterstützen?
Von 1 (keine Unterstützung) bis 5 (starke Unterstützung)

Antworten

3	3	5	4

Mittelwert $3,75$

Standardabweichung $0,83$

Frage 4

Frage Wie beurteilen Sie die Plausibilität der Prozessmodelle? Bedenken Sie dabei, dass nicht alle Aufrufe von Services geloggt werden.
Von 1 (gar nicht plausibel) bis 5 (komplett plausibel)

Antworten

4	3,5	4	4

Mittelwert $3,88$

Standardabweichung $0,22$

Frage 5

Frage Wie sehr unterstützt die textuelle Auflistung der Traces die Plausibilität?
Von 1 (gar nicht) bis 5 (sehr stark)

Antworten

3	5	3	2

Mittelwert $3,25$

Standardabweichung $1,09$

Frage 6

Frage Wie sehr unterstützt die visuelle Darstellung der Traces der Plausibilität?
Von 1 (gar nicht) bis 5 (sehr stark)

Antworten

5	5	5	1

Mittelwert $4,73$

Standardabweichung $1,73$

Frage 7

Frage Wie sehr hilft die visuelle Darstellung der Traces dem Verständnis des Modells?
Von 1 (gar nicht) bis 5 (sehr stark)

Antworten

5	5	5	5

Mittelwert 5

Standardabweichung 0

Frage 8

Frage Wie übersichtlich finden Sie die Darstellung der Graphen?
Von 1 (komplett unübersichtlich) bis 5 (sehr übersichtlich)

Antworten

4	3,5	4	5

Mittelwert $4,13$

Standardabweichung $0,54$

Frage 9

Frage Wie ansprechend finden Sie die interaktive Darstellung der kräftebasierten Layouts (Bewegung der Elemente in Echtzeit)?
Von 1 (sehr störend) bis 5 (sehr ansprechend)

Antworten

5	3,5	5	5

Mittelwert $5,63$

Standardabweichung $0,65$

Frage 10

Frage Denken Sie, dass bestehende Systeme die Funktionalität dieses Programms schon abdecken?
Von 1 (keine Funktionalität ist bereits abgedeckt) bis 5 (die Funktionalität ist vollständig abgedeckt)

Antworten

3	2	3	2

Mittelwert $2,5$

Standardabweichung $0,5$

Frage 11

Frage Wie schätzen Sie den Nutzen des Programmes ein, um Prozessabweichungen zu erkennen?
Von 1 (kein Nutzen) bis 5 (sehr großer Nutzen)

Antworten

4	5	4	5

Mittelwert $4,5$

Standardabweichung $0,5$

Frage 12

Frage Wie schätzen Sie den Nutzen des Programmes zur Dokumentation von Prozessen ein?
Von 1 (kein Nutzen) bis 5 (sehr großer Nutzen)

Antworten

4	4	4	5

Mittelwert $4,25$

Standardabweichung $0,43$

Frage 13

Frage Wie schätzen Sie den Nutzen des Programmes für die agile Entwicklung von Web Services?
Von 1 (kein Nutzen) bis 5 (sehr großer Nutzen)

Antworten

2	4	3	4

Mittelwert $3,5$

Standardabweichung $0,83$

Literaturverzeichnis

[ARW+07] W. M. van der Aalst, H. A. Reijers, A. J. Weijters, B. F. van Dongen, A. A. De Medeiros, M. Song, H. Verbeek. Business process mining: An industrial application. *Information Systems*, 32(5):713–732, 2007. (Zitiert auf den Seiten 40 und 42)

[AV08] W. M. van der Aalst, H. E. Verbeek. Process Mining in Web Services: The WebSphere Case. *IEEE Data Eng. Bull.*, 31(3):45–48, 2008. (Zitiert auf Seite 43)

[AWM04] W. Van der Aalst, T. Weijters, L. Maruster. Workflow mining: Discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1128–1142, 2004. (Zitiert auf den Seiten 30, 32 und 40)

[bea] BeanShell (Version vom 05.09.2015). URL https://en.wikipedia.org/wiki/BeanShell. (Zitiert auf Seite 63)

[BH86] J. Barnes, P. Hut. A hierarchical O (N log N) force-calculation algorithm. 1986. (Zitiert auf Seite 40)

[bpm11] BPMN 2.0, 2011. URL http://www.omg.org/spec/BPMN/2.0/. (Zitiert auf Seite 21)

[BS07] T. Bayer, D. M. Sohn. REST Web Services–. *Eine Einfuehrung (November 2002) http://www. oio. de/public/xml/rest-webservices. pdf*, 2007. (Zitiert auf Seite 54)

[CW98] J. E. Cook, A. L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 7(3):215–249, 1998. (Zitiert auf Seite 44)

[d3215] Force Layout, 2015. URL https://github.com/mbostock/d3/wiki/Force-Layout. (Zitiert auf Seite 40)

[DMV+05] B. F. van Dongen, A. K. A. de Medeiros, H. Verbeek, A. Weijters, W. M. Van Der Aalst. The ProM framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets 2005*, S. 444–454. Springer, 2005. (Zitiert auf den Seiten 37 und 45)

[Dwy01] T. Dwyer. Three dimensional UML using force directed layout. In *Proceedings of the 2001 Asia-Pacific symposium on Information visualisation-Volume 9*, S. 77–85. Australian Computer Society, Inc., 2001. (Zitiert auf Seite 44)

[ES15] A. E. Eiben, J. E. Smith. *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg, 2015. (Zitiert auf den Seiten 35 und 36)

[FR91] T. M. Fruchterman, E. M. Reingold. Graph drawing by force-directed placement. 1991. (Zitiert auf den Seiten 37 und 39)

[GV14] C. W. Günther, E. Verbeek. XES - Standard Definition, 2014. URL http://www.xes-standard.org/_media/xes/xesstandarddefinition-2.0.pdf. (Zitiert auf den Seiten 26 und 27)

[luc15] Apache Lucene, 2015. URL https://lucene.apache.org/core/. (Zitiert auf Seite 23)

[MAW03] A. K. A. de Medeiros, W. M. van der Aalst, A. Weijters. Workflow mining: Current status and future directions. In *On the move to meaningful internet systems 2003: Coopis, doa, and odbase*, S. 389–406. Springer, 2003. (Zitiert auf Seite 34)

[mbo11] Collapsible Force Layout, 2011. URL http://bl.ocks.org/mbostock/1062288. (Zitiert auf Seite 41)

[mbo12] Sticky Force Layout, 2012. URL http://bl.ocks.org/mbostock/3750558. (Zitiert auf Seite 41)

[MWA07] A. K. A. de Medeiros, A. J. Weijters, W. M. van der Aalst. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007. (Zitiert auf den Seiten 37 und 42)

[Pet62] C. A. Petri. Kommunikation mit automaten. 1962. (Zitiert auf Seite 14)

[pro15] Tutorial: Automating Process Mining with ProM's Command Line Interface, 2015. URL https://dirksmetric.wordpress.com/2015/03/11/tutorial-automating-process-mining-with-proms-command-line-interface/. (Zitiert auf Seite 63)

[Rei85] W. Reisig. *Petri nets: an introduction*, Band 4. Springer Science & Business Media, 1985. (Zitiert auf Seite 16)

[VDA11] W. Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer Science & Business Media, 2011. (Zitiert auf den Seiten 16, 17, 18, 19, 20, 24, 25, 26, 27, 28, 30, 33, 40, 44 und 51)

[VDA13] W. Van Der Aalst. Service mining: Using process mining to discover, check, and improve service behavior. *Services Computing, IEEE Transactions on*, 6(4):525–535, 2013. (Zitiert auf den Seiten 42 und 43)

[VDAAM+12] W. Van Der Aalst, A. Adriansyah, A. K. A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. van den Brand, R. Brandtjen, J. Buijs, et al. Process mining manifesto. In *Business process management workshops*, S. 169–194. Springer, 2012. (Zitiert auf den Seiten 52 und 73)

[WA03] A. Weijters, W. M. Van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003. (Zitiert auf Seite 40)

[WF94] S. Wasserman, K. Faust. *Social network analysis: Methods and applications*, Band 8. Cambridge university press, 1994. (Zitiert auf Seite 24)

[wik15a] Coulomsches Gesetz, 2015. URL https://de.wikipedia.org/wiki/Coulombsches_Gesetz. (Zitiert auf Seite 39)

[wik15b] Hookesches Gesetz, 2015. URL https://de.wikipedia.org/wiki/Hookesches_Gesetz. (Zitiert auf Seite 39)

Alle URLs wurden zuletzt am 20. 10. 2015 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift