Zur Eignung des Process Mining für die Überprüfung von Sicherheitsanforderungen in prozessgestützten Informationssystemen

Bestandsaufnahme und Fallstudien

Masterarbeit im Studiengang Angewandte Informatik

vorgelegt von

Meike Ullrich, BSc Kognitionswissenschaften

am 9. April 2012

Erstgutachter: Prof. Dr. Dr. h.c. Günter Müller

Betreuer: Dr. Rafael Accorsi

Institut für Informatik und Gesellschaft, Abteilung Telematik

Albert-Ludwigs-Universität Freiburg

Danksagung

Mein Dank gilt Prof. Dr. Dr. h.c. Günter Müller für die Übernahme der Begutachtung dieser Arbeit und darüber hinaus für eine Reihe von inspirierenden Vorlesungen und Seminaren am Institut für Informatik und Gesellschaft, die mein Interesse am Themengebiet des Instituts geweckt haben.

Besonderen Dank schulde ich meinem Betreuer Dr. Rafael Accorsi für seine Zuverlässigkeit, sein Engagement und das mir entgegengebrachte Vertrauen. Darüber hinaus habe ich ihm die spannende Themenstellung sowie wertvolle fachliche Diskussionen und vielseitige Denkanstöße zu verdanken.

Ebenfalls bedanke ich mich bei Thomas Stocker für seine Hilfsbereitschaft und die freundliche Einweisung zur Arbeit mit dem Simulationswerkzeug und bei Dr. Lutz Lowis für seine stetige Diskussionsbereitschaft.

Weiterhin gebührt mein Dank den vielen kooperativen Personen aus aller Welt, die meine Fragen rund um Process Mining und Simulation sowie zur praktischen Anwendung von Verfahren beantwortet haben: Michael Westergaard (TU Eindhoven), Anne Rozinat (Fluxicon), Andrea Burattin (University of Padua), Thomas Baier (HU Berlin) und Anne Baumgraß (Vienna University of Economics and Business) sowie weiteren Personen aus dem ProM-Forum[1].

Herzlicher Dank geht auch an Dipl.-Phys. Volker Then und Dipl.-Psych. Elisa Merkel für die gewissenhafte Durchsicht des Manuskripts.

An letzter Stelle – aber nicht zuletzt – danke ich meiner Familie für die unermüdliche Unterstützung: meinen Eltern und Schwiegereltern für das Verschaffen von ungestörten Zeitfenstern zur Verfassung dieser Arbeit, sowie meinem Mann Patrick und meiner Tochter Lina für das Besinnen aufs Wesentliche und den wunderbaren Ausgleich zur vielen Schreibtischarbeit.

[1] `http://www.win.tue.nl/promforum/`

Zusammenfassung

Die (teil-)automatisierte Abwicklung von Prozessen durch prozessgestützte Informationssysteme (PAIS) in großen Unternehmen führt zu einer wenig transparenten Prozessausführung. Daraus ergeben sich ernstzunehmende Risiken, wie z. B. Betrugsfälle, die durch interne Teilnehmer des Prozesses verursacht werden, sowie die Ausnutzung von Schwachstellen durch externe Angreifer. Die daher dringend notwendigen Kontrollmaßnahmen in Bezug auf Compliance und Sicherheit können jedoch bei gleichzeitig auftretendem Wunsch nach Flexibilität bei der Prozessausführung nur *ex post* stattfinden. Das Process Mining bietet eine Reihe von Verfahren, mit deren Hilfe die von PAIS während der Prozessausführung aufgezeichneten Ereignisprotokolle analysiert werden können. Inwieweit diese Verfahren einen Beitrag zur Compliance-Prüfung mit Schwerpunkt auf Sicherheitsanforderungen leisten können, wurde bisher nur unzureichend thematisiert.

Die vorliegende Arbeit verfolgt einen systematischen Ansatz zur Erstellung eines Katalogs von typischen Sicherheitsanforderungen, deren Einhaltung anschließend in zwei Fallstudien unter praktischer Anwendung von Process Mining-Verfahren überprüft wird. Dabei werden nicht nur Verfahren zur Konformitätsprüfung auf Ereignisprotokollen berücksichtigt, sondern auch Modelle, die durch Verfahren der Modell-Extraktion gewonnen werden, auf ihre Eignung als Grundlage für die Überprüfung untersucht.

Anhand der Fallstudien konnte unter Anwendung der Verfahren zur Konformitätsprüfung für die betrachteten Sicherheitsanforderungen demonstriert werden, dass sich (i) durch Betrachtung einzelner Instanzen Anforderungen bezogen auf Safety-Eigenschaften direkt auf dem Ereignisprotokoll überprüfen lassen. Die Möglichkeit, eine technische Einschränkung zu umgehen und darüber hinaus auch (ii) durch gleichzeitige Betrachtung mehrerer Instanzen Anforderungen bezogen auf Hypereigenschaften zu überprüfen, wurde ebenfalls umgesetzt und vorgestellt. Die Eignung von extrahierten Modellen als Grundlage für die Überprüfung von Anforderungen konnte nur eingeschränkt festgestellt werden. Bei der Anwendung konventioneller Verfahren zur Extraktion von Kontrollfluss-Modellen ist mit Informationsverlust zu rechnen, so dass Abweichungen von den untersuchten Anforderungen bezogen auf Kontrollfluss-Eigenschaften auf dieser Grundlage unerkannt bleiben.

Inhaltsverzeichnis

1. Einleitung

1.1. Hintergrund und Motivation

Getrieben durch technologischen Fortschritt und digitale Vernetzung nimmt der Einsatz von Informationssystemen zur computergestützten und automatisierten Verarbeitung von betrieblichen Prozessabläufen zu. Informationssysteme wie beispielsweise Workflow-Management Systeme (WfMS), Enterprise-Resource-Planning Systeme (ERP) oder Customer-Relationship-Management Systeme (CRM) bilden heutzutage das Rückgrat der meisten Organisationen. Sogenannte prozessgestützte Informationssysteme (*process-aware information systems*, PAIS) steuern und führen operationale Prozesse auf der Basis von Prozessmodellen aus [37] und tragen damit zu einer flexiblen und effizienten Prozessausführung bei.

Aufgrund des hohen Automatisierungsgrades von Prozessen entsteht jedoch eine neue Angriffsfläche, solange keine ausreichenden Kontrollen im Zusammenhang mit der Prozessausführung stattfinden. Dies ist in der Praxis häufig der Fall, da bei der Gestaltung von Prozessen überwiegend betriebswirtschaftliche Ziele im Vordergrund stehen und dabei klassische Sicherheitsschutzziele der zugrundeliegenden Informationstechnologie (IT) wie Vertraulichkeit und Integrität vernachlässigt werden [60]. Dabei existiert eine Reihe von Richtlinien und Vorschriften zur sicheren und korrekten Prozessausführung, die, zum Teil in Regularien wie z. B. dem deutschen Bundesdatenschutzgesetz (BDSG) oder dem amerikanischen Health Information Portability and Accountability Act (HIPAA) festgesetzt, für Unternehmen verpflichtend sind. Zum einen drohen bei Nicht-Einhaltung solcher gesetzlichen Vorschriften strenge Sanktionen. Zum anderen kann durch eine fehlerhafte oder mißbräuchliche Ausführung von Prozessen selbst materieller oder immaterieller Schaden entstehen [55]. Diese Gründe unterstreichen die Notwendigkeit, Lösungen für eine verbesserte Kontrolle und Überprüfung der Ausführung von Prozessen in Informationssystemen zu entwickeln, damit die Einhaltung der bestehenden Anforderungen (Compliance) gewährleistet werden kann.

Entsprechende Lösungsansätze zur Compliance-Prüfung lassen sich grundsätzlich auf verschiedenen Zeitebenen ansiedeln [55, 59]. *Vor der Ausführung* eines Prozesses (zur Entwurfszeit) kommen statische Analyseverfahren zum Einsatz. Mit dem Model Checking können beispielsweise alle möglichen Systemzustände

eines Modells auf bestimmte Eigenschaften überprüft werden (Verifikation) [20]. Über die Entwurfszeit hinausgehende Eigenschaften, die erst *während der Ausführung* (zur Laufzeit) feststehen, können mit der dynamischen Analyse, z. B. mit einem Compliance-Monitor [11], überwacht werden. Zuletzt kann *nach der Ausführung* (*a posteriori*) eine forensische Analyse (Audit) erfolgen. Basierend auf einer forensischen Analyse von aufgezeichneten Ereignissen der Prozessausführung können Evidenzen geniert werden, die als Indikatoren für Einhaltung oder Verletzung von Anforderungen dienen – nicht jedoch als formaler Beweis [10, 59].

Ein großer Vorteil der forensischen Analyse liegt darin begründet, dass bei *a posteriori* Betrachtung die Maximalmenge an Informationen über die Prozessausführung vorliegt. So kann beispielsweise auch die Einhaltung von verpflichtenden Sicherheitsvorschriften (Obligationen) überprüft werden. Allerdings können Compliance-Verletzungen zwar nachträglich ermittelt, jedoch nicht verhindert werden. Nichtdestotrotz wiegt der Wunsch nach weitestgehender Flexibilität bei der Prozessausführung in der Praxis schwer, denn Unternehmen müssen konstant auf sich verändernde Rahmenbedingungen, seien sie marktwirtschaftlicher, organisatorischer oder regulatorischer Natur, reagieren, um einen effektiven Betrieb zu gewährleisten [44]. In diesem Zusammenhang erfüllt nur die *a posteriori* Analyse die nötigen Voraussetzungen für die gewünschte Flexibilität, da sie keinerlei Einschränkungen für die Prozessausführung mit sich bringt.

In der Praxis spielen manuelle Auditverfahren noch immer eine große Rolle [27]. Diese allerdings erfordern nicht nur einen immensen Zeitaufwand verbunden mit hohen Kosten. Sie haben weitere gravierende Nachteile. Vielmals dienen Gegenstände wie z. B. Fragenkataloge als Grundlage für die Analyse. Dabei kann die notwendige Authentizität der auf diese Weise ermittelten Informationen nicht sichergestellt werden. In diesem Zusammenhang wird die Verfügbarkeit von durch PAIS aufgezeichneten Ereignisprotokollen bisher überwiegend ignoriert. Vorteile dieser Ereignisprotokolle als Grundlage für die forensische Analyse sind neben der automatisierten Dokumentation der Prozessausführung, dass sie (1) gegenüber reinen Transaktionsdaten auch prozessrelevante Metadaten enthalten (z.B. Zeitstempel, Teilnehmer, Ressourcen) und (2) diese nicht von den Teilnehmern am System manipuliert werden können [51]. Diese Vorteile bleiben jedoch aufgrund eines Mangels an spezialisierter Tool-Unterstützung ungenutzt. Audit-Berichte, die auf einer manuellen forensischen Analyse basieren, können bei einer auszugsweisen Betrachtung der großen zu untersuchenden Datenmengen zur Prozessausführung nicht den Anspruch auf Vollständigkeit erfüllen [27]. Es besteht die Gefahr, dass sicherheitsrelevante Vorfälle oder Abweichungen von Richtlinien im Prozessablauf unentdeckt bleiben.

Das Process Mining steht für eine Reihe von *a-posteriori* Verfahren, mit deren Hilfe das in Ereignisprotokollen durch PAIS dokumentierte Verhalten automatisiert analysiert werden kann. Dies dient der Gewinnung von Wissen über die Prozessausführung. In Anlehnung an das Data Mining aus dem Bereich des maschinellen Lernen entstanden, fokussiert das Process Mining statt der Daten- auf die Prozessebene, was sich mit dem zunehmend verschiebenden Interesse auf diese deckt [1]. Das Process Mining bietet dabei Methoden, um (a) zugrundeliegende *de facto* Prozessmodelle aus Ereignisprotokollen zu extrahieren (Modell-Extraktion), (b) die Übereinstimmung von Prozessmodellen oder Regeln mit dem Prozessablauf in Ereignisprotokollen zu prüfen (Konformitätsprüfung) und (c) bestehende Prozessmodelle um zusätzliche Informationen aus Ereignisprotokollen zu erweitern und zu verbessern (Erweiterung) [3]. Erste Untersuchungen [5, 12, 51, 53, 52] legen nahe, dass das Process Mining einen wertvollen Beitrag zu einer verbesserten Qualität der automatisierten Compliance-Prüfung leisten kann.

1.2. Problemstellung und Ziele

Bisher ist das Potenzial für die Compliance-Prüfung durch das Process Mining nicht erschöpfend beleuchtet worden. Es existiert keine systematische Gegenüberstellung von Anforderungen zur Prozessausführung und Process Mining-Verfahren, die für deren Überprüfung geeignet sind. Weitestgehend unberücksichtigt ist der in [5, 12, 51] vorgeschlagene und diskutierte Einsatz von extrahierten *de facto* Prozessmodellen als Basis für eine Compliance-Prüfung, bzw. für die Evidenzgenerierung.

Demnach besteht das Hauptziel der Arbeit darin, die bisherigen Erkenntnisse über die Eignung von Process Mining für die Compliance-Prüfung zu erweitern. Im Vordergrund steht die Frage, welche Anforderungen tatsächlich für die Analyse mit Process Mining-Verfahren greifbar sind. Basierend auf dieser Ausgangsfrage werden konkrete Antworten auf die folgenden Teilfragen gesucht:

1. Welche Anforderungen lassen sich praktisch mit bestehenden Process Mining-Verfahren der Konformitätsprüfung kontrollieren? Wo liegen dabei Grenzen des Process Minings?
2. Sind durch die Modell-Extraktion des Process Minings gewonnene *de facto* Prozessmodelle als Basis für die Überprüfung von Anforderungen geeignet?

Compliance-Anforderungen können vertraglicher, gesetzlicher oder regulatorischer Natur sein. In dieser Arbeit liegt der Schwerpunkt auf der Betrachtung

von Anforderungen, die der Sicherheit von Prozessen dienen. Es werden also Sicherheitsanforderungen betrachtet.

1.3. Ansatz

Der Ansatz dieser Arbeit lässt sich in einen konzeptionellen und einen praktischen Teil gliedern. Der konzeptionelle Teil befasst sich mit den zu untersuchenden Gegenständen und wird von folgenden Bausteinen getragen:

(1) Erstellung eines *Katalogs von typischen Sicherheitsanforderungen*, die für Prozesse insbesondere in PAIS gelten. Dazu wird die methodische Anforderungsanalyse von Müller et al. [61] unter Berücksichtigung weiterer Quellen zum Thema Prozesssicherheit und Compliance angewendet.

(2) Ausarbeitung einer *Übersicht von Process Mining-Verfahren* der Typen Modell-Extraktion und Konformitätsprüfung durch eine entsprechende Literaturrecherche.

Die beiden Bausteine bilden die Grundlage für den praktischen Teil der Arbeit, welcher die Durchführung von Fallstudien verschiedener Prozesse behandelt. Dabei werden zu den Sicherheitsanforderungen aus (1) geeignete Fallbeispiele entworfen und mit Process Mining-Verfahren aus (2) analysiert. Die Organisation der Fallstudien orientiert sich an den Richtlinien von Runeson u. Höst [68] und wird in die folgenden Schritte gegliedert:

1. **Entwurf:** Auswahl der Fallbeispiele und Planung der folgenden Phasen.
2. **Vorbereitung:** Simulation von Ereignisprotokollen mit darauffolgender Einbettung von Abweichungen.
3. **Durchführung:** Anwendung von Process Mining-Verfahren auf diesen Ereignisprotokollen zur Evidenzsammlung und anschließende Analyse der Ergebnisse.

Die Fallstudien dienen auf diese Weise als Methode, um für die Compliance-Prüfung mit Process Mining entweder (i) einen entsprechenden Machbarkeitsnachweis zu liefern oder (ii) Schwächen und Grenzen der Verfahren praktisch aufzuzeigen. Gleichzeitig können Verbesserungsvorschläge und Forschungslücken aufgezeigt werden.

1.4. Aufbau der Arbeit

Kapitel 2 ist einer Beschreibung der Ausgangsgrundlagen der Arbeit gewidmet. Hier wird neben einer Einführung relevanter Begriffe und der Prozessmodellierung mit Petri-Netzen auch ein Katalog von Sicherheitsanforderungen erarbeitet. Die Grundlagen des Process Minings werden in Kapitel 3 zusammengefasst dargestellt. Es folgt die Beschreibung des Entwurfs der Fallstudien in Kapitel 4 sowie deren Durchführung in Kapitel 5. In Kapitel 6 werden die Ergebnisse und Validität der Fallstudien diskutiert sowie verwandte Arbeiten betrachtet. Kapitel 7 präsentiert abschließend Zusammenfassung und Ausblick.

2. Prozesse und Sicherheit

Dieses Kapitel beschäftigt sich mit Prozessen, insbesondere solchen wie sie in prozessgestützten Informationssystemen (PAIS) auftreten, und deren Sicherheit. Es beginnt zur Beschreibung der relevanten Grundlagen dieser Arbeit mit einer Definition der Begriffe *Prozess*, *Workflow* und *PAIS*, die an einem Beispiel erläutert werden (Abschnitt 2.1). Im nächsten Abschnitt 2.2 liegt der Fokus auf Prozessmodellen. Neben der Beschreibung des Einsatzbereichs von Prozessmodellen wird gezeigt, wie Prozessmodelle auf Basis von Petri-Netzen umgesetzt werden können. Schließlich wird zum Thema Sicherheit in Abschnitt 2.3 ein Katalog von typischen Sicherheitsanforderungen für Prozesse in PAIS erarbeitet, der im weiteren Verlauf der Arbeit für die Auswahl von Process Mining-Verfahren und den Entwurf der Fallbeispiele eine tragende Rolle spielt.

2.1. Begriffsdefinitionen

Es werden zunächst grundlegende Begriffe eingeführt. Im Folgenden wird auf die Definitionen von van der Aalst u. Stahl [8] zurückgegriffen.

Definition 1 (Geschäftsprozess)**.** Ein Geschäftsprozess besteht aus einer Menge von Aktivitäten, die in einer organisatorischen und technischen Umgebung ausgeübt werden. Diese Aktivitäten sind so koordiniert, dass sie gemeinsam ein Geschäftsziel verwirklichen. [. . .] ([8, S. 4])

Über diese Definition hinaus werden nach Hansen u. Neumann [45] Geschäftsprozesse, die teilweise oder vollständig automatisiert sind, auch als *Workflow* bezeichnet. Im folgenden Text soll der Begriff *Prozess* den in der Fachliteratur verbreiteten Terminus *Geschäftsprozess (business process)* ablösen, da dieser eine besondere Beziehung zum Bereich der kommerziellen, erwerbsorientierten Wirtschaft suggeriert. Eine solche Einschränkung ist jedoch im Rahmen dieser Arbeit und auch für das Process Mining unerheblich.

Abbildung 2.1 zeigt ein einfaches Beispiel für einen Prozess zur Abwicklung einer Behandlung, wie er bei einer ambulanten Behandlung in einer Arztpraxis eingesetzt werden könnte. Die erste Aktivität ist die *Aufnahme* des Patienten, der angibt, welche Leistung (z. B. IGEL, individuelle Gesundheitsleistung) er

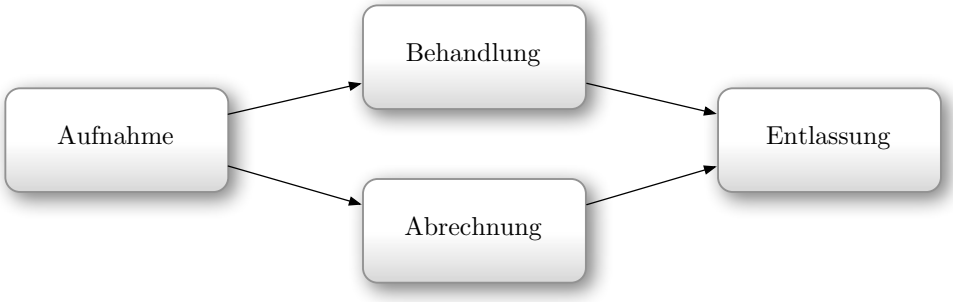

Abbildung 2.1.: Beispiel eines Prozesses zur Abwicklung einer Behandlung.

in Anspuch nehmen möchte. Danach folgen die Aktivitäten *Behandlung* und *Abrechnung*, anschließend wird der Patient entlassen (Aktivität *Entlassung*). Das Ziel dieses Prozesses ist es, sowohl die Behandlung als auch die Abrechnung sicherzustellen, was durch die gegebene Koordination von Aktivitäten erreicht wird.

Definition 2 (Prozessgestütztes Informationssystem)**.** Ein prozessgestütztes Informationssystem (*process-aware information system*, PAIS) ist ein Softwaresystem, welches operationale Prozesse [...] auf der Basis von Geschäftsprozessmodellen verwaltet und ausführt. ([8, S. 49])

Klassische Beispiele für PAIS sind Workflow-Management-Systeme (WfMS) und Business-Process-Management (BPM) Systeme. Diese Systeme unterstützen operationale Prozesse und werden durch eine explizite Prozessrepräsentation gesteuert [1]. Der Unterschied zwischen Workflow Management (WfM) und BPM kann dabei anhand des BPM-Lebenszyklus verdeutlicht werden (siehe Abb. 2.2). Der BPM-Lebenszyklus beschreibt die verschiedenen Phasen der Unterstützung von operationalen Prozessen. In der Design-Phase werden Prozesse entworfen oder angepasst. Die Implementierung und Umsetzung dieser Prozesse findet in der Konfigurationsphase durch die Konfiguration eines PAIS, z. B. eines WfMS statt. Anschließend beginnt in der Enactment-Phase die Prozessausführung durch das zuvor konfigurierte System. In der Diagnose-Phase wird die Prozessausführung analysiert, um Probleme zu identifizieren und gegebenenfalls eine Anpassung des Prozessentwurfs zu veranlassen. Beim WfM liegt der Fokus auf der unteren Hälfte des BPM-Lebenszyklus. Das BPM erweitert den traditionellen WfM-Ansatz um eine verbesserte Unterstützung der Diagnose-Phase, allerdings sind Werkzeuge zur Diagnose von Prozessen nach dem aktuellem Stand nach wenig ausgereift [6].

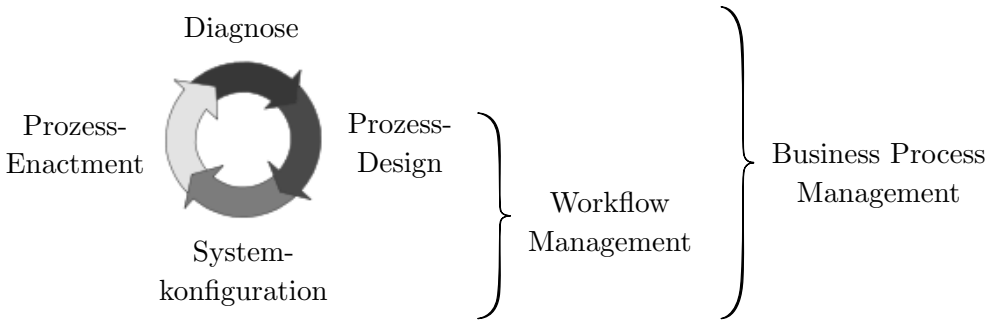

Abbildung 2.2.: Vergleich des Workflow Management und Business Process Management anhand des BPM-Lebenszyklus. Modifiziert nach van der Aalst et al. [6]

Der Beispielprozess zur Behandlungsabwicklung ist durch die Darstellung in Abb. 2.1 nur unzureichend repräsentiert und in dieser Form nicht für die Konfiguration eines PAIS geeignet. Eine Lösung bietet die Verwendung von formalen Prozessmodellen mit einer wohldefinierten Semantik, wie sie im nächsten Abschnitt vorgestellt werden.

2.2. Prozessmodelle

In dieser Arbeit nehmen Prozessmodelle eine zentrale Stellung ein. Zum einen dienen sie als Ausgangsgrundlage für die Fallbeispiele und zum anderen spielen sie, wie später gezeigt, eine große Rolle im Zusammenhang mit Process Mining-Verfahren. Die modellhafte Darstellung eines Prozesses (Prozessmodellierung) bietet Vorteile, die an dieser Stelle kurz aufgelistet werden sollen. Prozessmodelle können nach van der Aalst u. Stahl [8, S. 45] zu folgenden Zwecken eingesetzt werden:

(1) Einblick: beim Entwurf von Prozessmodellen müssen Strukturen und Regeln von Prozess und Informationssystem explizit gemacht werden, so dass dabei sowohl Verbesserungsmöglichkeiten, als auch Fehler aufgezeigt werden können.

(2) Analyse: Prozessmodelle dienen als Ausgangsbasis für verschiedene Formen der Analyse, z. B. Validierung, Verifikation oder Performanz-Analyse, darüber hinaus auch für die Simulation.

(3) Realisierung: Prozessmodelle dienen als Spezifikation für die Ausführung von Informationssystemen.

Dies macht deutlich, dass Prozessmodelle weit über ihre Rolle zur Systemkonfiguration im Rahmen von PAIS hinaus von Bedeutung sind.

Für die Modellierung von Prozessen existiert eine Vielzahl von Notationen und Modellierungssprachen wie z. B. der Business Process Modeling Notation (BPMN), der Business Process Execution Language (BPEL) oder Yet Another Workflow Language (YAWL). Da diesen jedoch eine für die Verifikation geeignete Semantik fehlt, sollen hier Petri-Netze betrachtet werden. Petri-Netze bieten eine intuitive grafische Notation nebst einer fundierten formalen Grundlage und zählen darüber hinaus zu den ältesten und am besten erforschten Prozessmodellierungssprachen.

Definition 3 (Petri-Netz)**.** Ein Petri-Netz ist ein Tripel $N = (S, T, F)$ wobei S eine endliche Menge von Stellen und T eine endliche Menge von Transitionen mit $S \cap T = \varnothing$ darstellt. F ist eine Menge von gerichteten Kanten, die Flussbeziehungen, für die $F \subseteq (S \times T) \cup (T \times S)$ gilt. Ein markiertes Petri-Netz ist ein Paar (N, M), wobei $N = (S, T, F)$ ein Petri-Netz und $M \in \mathbb{B}(S)$ eine Multimenge[1] über S ist, welche die Markierung beschreibt. Die Menge aller markierten Petri-Netze wird mit $\mathcal{N}$ bezeichnet. ([3, S. 33])

Diese Definition beschreibt die einfachste Form von Petri-Netzen, auch *low-level* Petri-Netze genannt. Nach dieser ist ein Petri-Netz ein bipartiter Graph, d. h. ein Modell für die Beziehungen zwischen Stellen und Transitionen (daher auch S/T-Netz genannt). Die Netzstruktur ist statisch, in markierten Petri-Netzen können jedoch sogenannte *Marken* gesteuert über die *Feuerregel* durch das Netz fließen. Hierbei wird eine Marke in einer initialen Stelle des Netzes positioniert. Eine Transition wird *aktiviert*, wenn sich in jeder ihr vorangehenden Stelle eine Marke befindet. Eine aktivierte Transition kann feuern, indem sie jeweils eine Marke aus ihren vorangehenden Stellen konsumiert und jeweils eine Marke in auf sie folgenden Stellen positioniert.

Das Beispiel für ein Prozessmodell in Form eines markierten Petri-Netzes in Abb. 2.3 greift den Prozess aus Abschnitt 2.1 in erweiterter Form wieder auf. Die Transitionen a, b, c, d und e repräsentieren hierbei die Aktivitäten $Aufnahme$, $Behandlung$, $Abrechnung\ GKV$ sowie $Abrechnung\ PKV$ (gesetzliche oder private Krankenversicherung) und $Entlassung$. Die Abfolge dieser Aktivitäten wird durch die vorhandenen Stellen und Kanten über die Anwendung der Feuerregel gesteuert. Im diesem Beispiel wird ausgehend von der Markierung $M = [start]$ die Transition a ($Aufnahme$) aktiviert. Diese konsumiert die Marke

[1] $\mathbb{B}(D) = D \to \mathbb{N}$ ist die Menge aller Multimengen über einer endlichen Domain D, so dass die Multimenge $M \in \mathbb{B}(S)$ für jede Stelle $s \in S$ mit $M(s)$ angibt, wie oft s in der Multimenge auftritt. Dadurch wird die Anzahl von Marken $M(s)$ in jeder Stelle s des Netzes beschrieben.

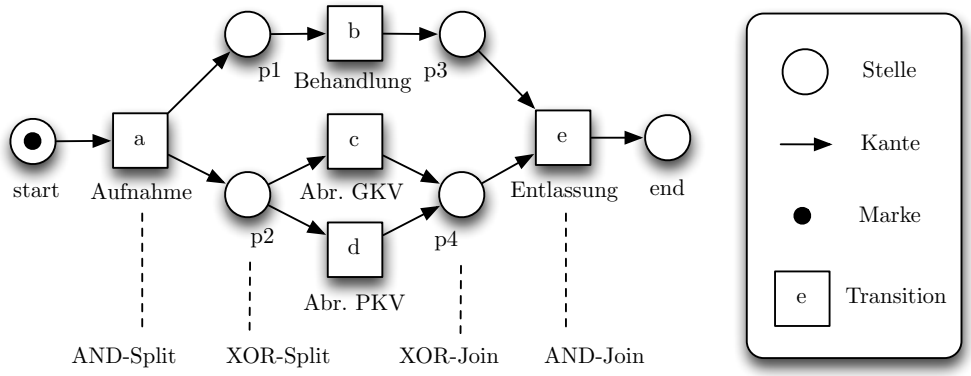

Abbildung 2.3.: Petri-Netz basiertes Prozessmodell; Marke in Stelle *start*, bzw. Markierung $M = [start]$.

in Stelle *start* und positioniert jeweils eine Marke in den darauffolgenden Plätzen $p1$ und $p2$, so dass $M = [p1, p2]$ gilt. Durch diese Verzweigung in zwei parallele Folgepfade ergibt sich ein sogenannter *AND-Split*. Auf diese Weise lassen sich mit Petri-Netzen nebenläufige Aktivitäten repräsentieren. Auf Stelle $p2$ folgen die zwei Transitionen c (*Abrechnung GKV*) und d (*Abrechnung PKV*), so dass nur eine dieser beiden aktiviert werden kann. Ein solches *XOR-Split* entspricht dem logischen exklusiven OR und repräsentiert die wahlweise Ausführung einer von mehreren Aktivitäten im Prozess.

Neben den im Beispiel gezeigten Konzepten nebenläufiger (AND-Split/Join) und wahlweiser Ausführung (XOR-Split/Join) lassen sich mit Petri-Netzen darüber hinaus auch sequentielle (Ketten aus Stellen und Transitionen) sowie iterative Ausführung (Schleifen) realisieren (siehe Abb. 2.4). Die Verkettung einiger Ausführungsmuster macht den Einsatz von sogenannten versteckten Transitionen (*hidden* oder *silent transitions*) erforderlich. Soll beispielsweise auf eine Aktivität A entweder die Aktivität B oder die beiden parallelen Aktivitäten C und D folgen, so lässt sich die Verkettung von einem XOR-Split zu einem AND-Split nur unter Verwendung einer versteckten Transition realisieren, wie in Abb. 2.5 gezeigt. Die versteckte Transition bildet dabei keine Aktivität des Prozesses ab, sondern dient nur zur Erstellung des vorgesehenen Pfades.

Zusammengenommen bieten Petri-Netze damit eine breite Auswahl an typischen Ausführungsmustern, die zur Modellierung der Reihenfolge von Aktivitäten im Prozessablauf (Kontrollfluss) eines Großteils von realen Prozessen genügt.

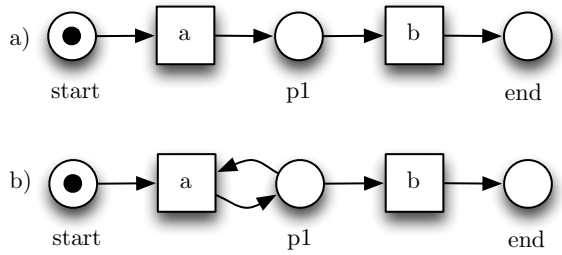

Abbildung 2.4.: Petri-Netz Ausführungsmuster: a) sequentielle Ausführung, b) iterative Ausführung.

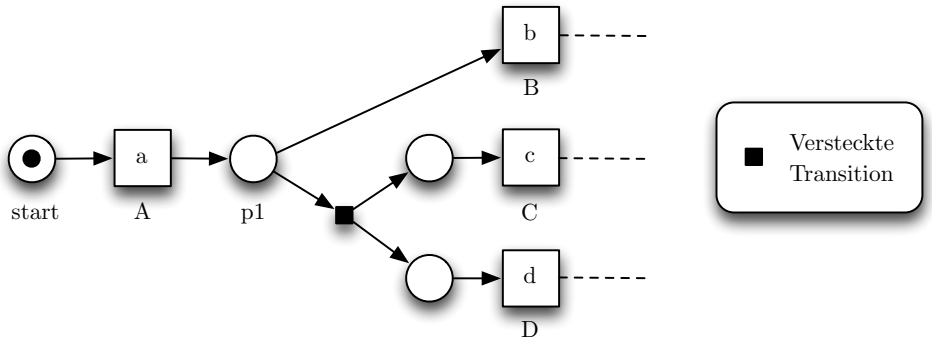

Abbildung 2.5.: Beispiel für ein Petri-Netz-Prozessmodell mit einer versteckten Transition.

2.3. Sicherheitsanforderungen für Prozesse in PAIS

In diesem Abschnitt wird ein Katalog mit Beispielen für Sicherheitsanforderungen im Zusammenhang mit der Ausführung von Prozessen in PAIS erarbeitet. Dazu wird die von Müller et al. [61] entworfene methodische Anforderungsanalyse angewendet, um auf Grundlage der Betrachtung von (1) Prozessaspekten und (2) Sicherheitsfacetten entsprechende Anforderungen aufzustellen[2]. Abbildung 2.6 zeigt diese beiden Gegenstände, wie sie während der Anforderungsanalyse von Müller et al. [61] gegenübergestellt werden. Basierend auf den Konzepten

[2]Müller et al. [61] demonstrieren die methodische Anforderungsanalyse an einem konkreten Fallbeispiel, wohingegen hier mit ihrem Ansatz von einem einzelnen Fallbeispiel unabhängige Anforderungen aufgestellt werden.

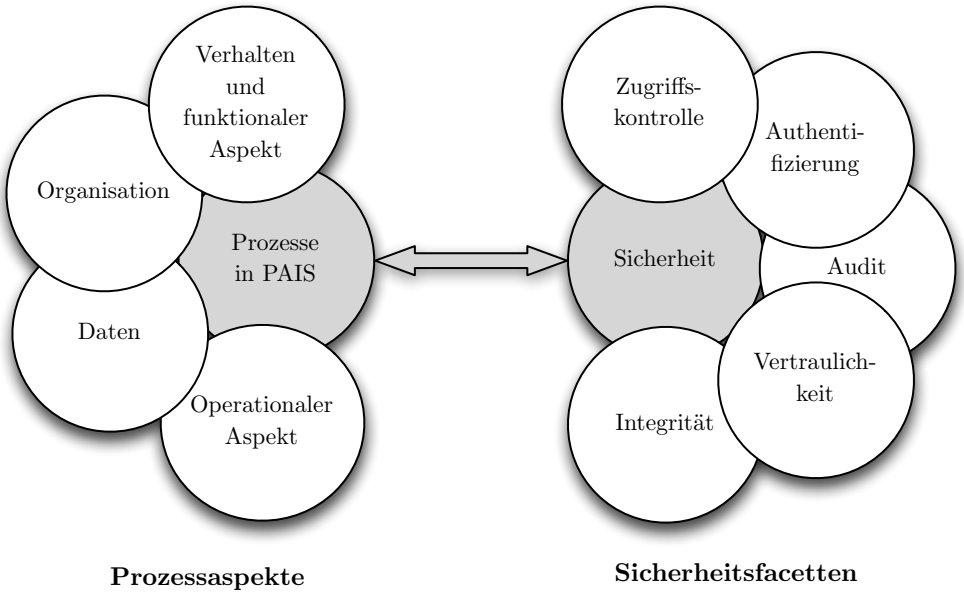

Abbildung 2.6.: Gegenstände der methodischen Anforderungsanalyse zur Aufstellung von Sicherheitsanforderungen für Prozesse in PAIS, vgl. [73, S. 6].

von Charfi u. Mezini [28], Jablonski u. Bussler [50] können für einen Prozess die folgenden Aspekte betrachtet werden:

- **Verhalten und funktionaler Aspekt** (*behavioural and functional aspect*): Kontrollfluss mit kausalen und temporalen Beziehungen sowie die strukturelle Komposition von Prozessen (Aktivitäten und Muster)
- **Organisation** (*organizational aspect*): Organisatorische Strukturen und Teilnehmer am Prozess
- **Daten** (*informational aspect*): Verwendung von Daten und Datenfluss im Prozess
- **Operationaler Aspekt** (*operational aspect*): Einbindung von externen Anwendungen oder Mechanismen in einen Prozess

Diese Aspekte sind zwar anhand ihrer Definition klar voneinander getrennt, Anforderungen zur Sicherheit umfassen allerdings in der Regel oft mehr als

einen Aspekt gleichzeitig[3]. Die für jeden Prozessaspekt in Frage kommenden Sicherheitsfacetten Zugriffskontrolle, Authentifizierung, Audit, Vertraulichkeit und Integrität sind Bestandteil der ITU-T Recommendation X.810 [49] und demnach essentiell für die Sicherheit von IT-Anwendungen.

Bei der im Folgenden vorgestellten Anforderungsanalyse werden die Prozessaspekte der Reihe nach in jeweils eigenen Absätzen betrachtet und dabei die verschiedenen Sicherheitsfacetten eingebracht. Die Entwicklung von konkreten Beispielen für Sicherheitsanforderungen in einem Anforderungskatalog beruht in dieser Arbeit nicht nicht nur auf Grundlage der von Müller et al. [61] vorgestellten Sicherheitsfacetten. Es werden darüber hinaus weitere Quellen zum Thema Prozess- bzw. Workflow-Sicherheit und Compliance (Atluri u. Warner [19], Lowis [55], Müller et al. [59], Workflow Management Coalition [78]) sowie auch verwandte Fallbeispiele (Accorsi u. Stocker [12]) berücksichtigt.

Verhalten und funktionaler Aspekt. Für die sichere Ausführung von Prozessen ist die Einhaltung von Anforderungen auf der Kontrollfluss-Ebene fundamental. Zunächst kann das Enthaltensein von Aktivitäten im Prozess festgelegt werden, z. B. Aktivität *Patientendaten löschen* darf nicht im Prozess enthalten sein [55]. Dies entspricht Anforderung **(A1a)** in Tabelle 2.1, die entgegengesetzte Ausprägung wird durch Anforderung **(A1b)** beschrieben.

Auch die Reihenfolge von Aktivitäten im Prozessablauf spielt eine wichtige Rolle. Die klassische Zugriffs- oder auch Zugangskontrolle erhebt die Bedingung, dass vor Ausführung einer Aktivität eine andere Aktivität stattgefunden haben muss, beispielsweise *Kundendaten einholen* vor *Kontoeröffnung*. Das Konzept der Nutzungskontrolle erweitert die Zugangskontrolle um die Berücksichtigung von Obligationen [62], also Anforderungen welche die Durchführung zukünftiger Aktivitäten ab dem Zugangszeitpunkt betreffen, z. B. muss nach der *Kontoeröffnung* eine *Verifikation der Kundenidentität* stattfinden [59]. Somit kann die Ausführung einer Aktivität an eine Bedingung, siehe **(A2a)**, oder eine Obligation, siehe **(A2b)**, geknüpft sein.

Bei einer besonders restriktiven Form von Anforderungen wie **(A3)** werden alle möglichen Reihenfolgen von Aktivitäten im Prozessablauf festgelegt. Diese können z. B. durch ein *de jure* Prozessmodell vorgegeben sein.

[3] Dies drückt sich auch in der Umsetzung von Prozessen durch die gängigen Prozessmodellierungssprachen aus. Kontrollmechanismen für Anforderungen mit Bezug auf mehrere Aspekte erstrecken sich häufig über über mehrere Prozess-Spezifikationen und eine Modularisierung ist nicht möglich. Aus diesem Grund ist es oft sehr umständlich, anhand der Prozess-Spezifikation nachzuvollziehen, ob Anforderungen entsprechend berücksichtigt werden [28]. Auch dies spricht für eine *a posteriori* Überprüfung von Anforderungen wie sie in dieser Arbeit vorgenommen wird.

Tabelle 2.1.: Sicherheitsanforderungen zu Verhalten und zum funktionalen Aspekt.

Name	Beschreibung
(A1a)	Aktivität X darf nicht enthalten sein.
(A1b)	Aktivität X muss enthalten sein.
(A2a)	Wird Aktivität X ausgeführt, muss zuvor Aktivität W ausgeführt worden sein.
(A2b)	Wird Aktivität X ausgeführt, muss danach Aktivität Y ausgeführt werden.
(A3)	Alle Prozessabläufe entsprechen vorgegebenen Reihenfolgen von Aktivitäten, die z. B. durch ein Prozessmodell definiert werden.
(A4)	Auf einen Entscheidungspunkt folgt Aktivität X falls Bedingung B wahr ist, Aktivität Y anderenfalls.
(A5)	Zwischen Aktivität X und Aktivität Y muss eine endliche Zeitspanne Z liegen.

Ist im Kontrollfluss des Prozessablaufs ein Entscheidungspunkt vorhanden wie es bei einer wahlweisen Ausführung (XOR-Split, siehe Abschnitt 2.2) der Fall ist, so muss in Abhängigkeit bestimmter Umstände zur Prozessausführung über die Reihenfolge entschieden werden. Beispielsweise muss ein Kreditvertrag ab einem Kreditwert von über $50.000 an einen Manager zur Unterschrift übermittelt werden, ansonsten nicht [12]. Anforderungen wie **(A4)** können solche Bedingungen an Entscheidungspunkten repräsentieren.

Der verhaltensbezogene und funktionale Aspekt umfasst auch die temporalen Beziehungen zwischen Aktivitäten. Eine entsprechende Anforderung wie **(A5)** lässt sich z. B. in dem Fallbeispiel aus [12] finden. Dort wird gefordert, dass nach Bestätigung einer Kreditvergabe innerhalb von 7 Tagen ein Vertrag vorbereitet wird.

Organisation. Unbestreitbar ist die Zugriffskontrolle die wichtigste Sicherheitsfacette im Zusammenhang mit dem Organisationsaspekt von Prozessen. Im Rahmen der Zugriffskontrolle wird festgelegt, ob ein Teilnehmer eine Aktivität des Prozesses ausführen darf, woraus sich die grundlegende Anforderung **(A6a)** ergibt (siehe Tabelle 2.2). Große Organisationen setzen häufig Konzepte rollenbasierter Zugriffskontrolle (*role-based access control*, RBAC) ein, um durch die Zuordnung von Rollen Teilnehmer mit gleichen Berechtigungen effektiv verwalten zu können [38]. Daraus wird die restriktive Anforderung **(A6b)** abgeleitet.

Neben der Zugriffskontrolle werden oft Bedingungen zur sogenannten Funktionstrennung (*separation* oder *segregation of duty*, SoD) festgelegt [19]. Mit Hilfe solcher internen Kontrollen soll das Betrugsrisiko durch missbräuchliche

Tabelle 2.2.: Sicherheitsanforderungen zum Organisationsaspekt.

Name	Beschreibung
(A6a)	Aktivität X darf nur von einem autorisierten Teilnehmer T ausgeführt werden.
(A6b)	Alle Teilnehmer müssen die Berechtigungen der RBAC-Vorgaben einhalten.
(A7a)	Führt Teilnehmer $T1$ die Aktivität X aus, so muss die Aktivität Y in der *gleichen Instanz* des Prozesses von einem Teilnehmer $T2$ ausgeführt werden, so dass $T1 \neq T2$ gilt.
(A7b)	Führt Teilnehmer $T1$ die Aktivität X aus, so muss die Aktivität Y in einer *anderen Instanz* des Prozesses von einem Teilnehmer $T2$ ausgeführt werden, so dass $T1 \neq T2$ gilt.

Ausführung von Prozessaktivitäten gemindert werden. Dies geschieht durch die Einschränkung, dass ein Teilnehmer, der eine Aktivität (z. B. *Scheck vorbereiten*) ausführt, eine andere Aktivität (z. B. *Scheck einlösen*) nicht ausführen darf (obwohl er dies den klassischen Zugriffskontrollrechten nach dürfte). In [19] wird zwischen Intra-Instanz- und Inter-Instanz-SoD unterschieden, so dass einmal Aktivitäten innerhalb einer Instanz und einmal Aktivitäten zwischen verschiedenen Instanzen von der Einschränkung betroffen sind. Daraus ergeben sich die Anforderungen **(A7a)** und **(A7b)**. Parallel zur Funktionstrennung gibt es auch das Konzept der Funktionsbindung (*binding of duty*, BoD), wobei hier entgegengesetzt festgelegt ist, dass ein Teilnehmer, der eine Aktivität ausführt, auch eine andere Aktivität des Prozesses ausführen muss. Aufgrund der Ähnlichkeit der beiden Konzepte genügt es, die Anforderungen zur Funktionstrennung zu betrachten. Somit wird auf eine Aufnahme von Anforderungen zur Funktionsbindung in den Anforderungskatalog verzichtet.

Daten. Nicht nur für den Organisationsaspekt, sondern auch bei Fokussierung auf Daten im Prozess lassen sich im Zusammenhang mit der Sicherheitsfacette Zugriffskontrolle relevante Anforderungen aufstellen. Zugriffsrechte werden aus dieser Perspektive von der reinen Zuordnung von Teilnehmern zu Aktivitäten auf eine Zuordnung zu Daten erweitert. Ein Zugriff auf Daten im Prozess findet zwar über die Ausführung einer Aktivität statt, ist jedoch auf bestimmte Daten eingeschränkt, siehe Anforderung **(A8)**, welche Anforderung **(A6a)** zur Zugriffskontrolle des Organisationsaspekts um den Datenaspekt erweitert (siehe Tabelle 2.3).

Im Zusammenhang mit den Sicherheitsfacetten Vertraulichkeit und Zugriffskontrolle kann für den Datenaspekt der sogenannte Interessenkonflikt angeführt werden. Ein solcher kann entstehen, wenn innerhalb eines Prozesses Daten

Tabelle 2.3.: Sicherheitsanforderungen zum Datenaspekt.

Name	Beschreibung
(A8)	Aktivität X mit Zugriff auf Daten D darf nur von einem autorisierten Teilnehmer T ausgeführt werden.
(A9)	Alle Teilnehmer die an der Ausführung einer Prozessinstanz beteiligt sind, müssen zur gleichen Partei gehören.
(A10)	Prozessaktivitäten, die Eingabe/Ausgabe-Operationen realisieren, dürfen bei der Ausführung durch Teilnehmer verschiedener Parteien keine Daten als Eingabe verwenden, die von einer anderen Partei erzeugt, bzw. ausgegeben wurden.
(A11)	Teilnehmer am Prozess dürfen nicht ableiten können, welche Bedingungen über Daten die Prozessausführung beeinflussen, z. B. bei einer automatisch durchgeführten Berechnung auf Grundlage von Daten (Isolation).

verschiedener Parteien (z. B. Abteilungen mit unterschiedlichen Zielen) behandelt werden [19]. Eine einfache Form der sogenannten Chinese Wall-Richtlinie kann vorgeben, dass Parteien völlig voneinander getrennt agieren müssen. Die entsprechende Anforderung **(A9)** legt fest, dass nur Teilnehmer der gleichen Partei gemeinsam einen Prozess ausführen können. So dürfen die jeweiligen Parteien zwar den gleichen Prozess verwenden, einzelne Prozessinstanzen bzw. Fälle können jedoch nur von einer einzelnen Partei bearbeitet werden.

Nach Denning u. Denning [32] ist es das Ziel der Informationsflusskontrolle, die Verbreitung von Informationen zwischen Objekten innerhalb eines Systems zu regulieren. Eine Informationsfluss-Richtlinie spezifiziert eine Menge von Sicherheitsklassen für Informationen, eine Flussbeziehung mit erlaubten Flüssen zwischen diesen Klassen und eine Methode, um jedes Speicherobjekt einer bestimmten Sicherheitsklasse zuzuweisen. Während der Systemausführung kann ein Informationsfluss von einem Objekt x zu einem Objekt y enstehen, wenn Operationen angewendet werden, bei denen der Wert von x verwendet wird, um einen Wert für y abzuleiten. Dabei wird zwischen einem *expliziten Informationsfluss* (auch Datenfluss genannt) und einem *impliziten Informationsfluss* unterschieden. Beim expliziten Informationsfluss von Objekt x zu Objekt y beeinflusst der Wert von x direkt den Wert y, z. B. bei einem Funktionsaufruf $f(x) = y$ oder einer Eingabe/Ausgabe-Operation, bei der x gelesen und y geschrieben wird. Ein impliziter Informationsfluss von Objekt x zu Objekt y findet statt, wenn der Wert von x den Wert von y nur indirekt beeinflusst, wie dies z. B. typischerweise für eine Bedingung der Form `if (x == 0) then y = z;` der Fall ist. Anforderungen **(A10)** und **(A11)** stellen Beispiele für Richtlinien

zum expliziten bzw. impliziten Informationsfluss im Zusammenhang mit der Prozessausführung dar.

Operationaler Aspekt. Der operationale Aspekt von Prozessen berücksichtigt die Einbindung von externen Anwendungen und Mechanismen. In diesem Zusammenhang werden die in den vorangegangenen Absätzen noch nicht betrachteten Sicherheitsfacetten Authentifizierung und Audit vorgestellt.

Der Workflow Management Coalition [78] nach müssen Teilnehmer vor Aufnahme einer Prozessaktivität authentifiziert werden. Die Authentifizierung als Mechanismus zur Identitätsfeststellung ist eine wichtige Voraussetzung für die Zugriffskontrolle. Ist die Authentifizierung als eigenständige Aktivität im Prozess repräsentiert, kann mit der bereits zuvor aufgestellten Anforderung **(A2a)** überprüft werden, ob vor einem Zugriff eine Authentifizierung erfolgt ist. Dabei ist jedoch zu beachten, dass auf diese Weise nicht kontrolliert werden kann, ob der Authentifizierungsmechanismus effektiv ist, d. h. ob er korrekt umgesetzt oder implementiert wurde. Eine solche Prüfung ist auf Prozessebene nicht möglich und muss direkt auf der Anwendungs- bzw. Quelltext-Ebene erfolgen.

Die Forderung der Workflow Management Coalition [78] nach der Aufzeichnung von Audit-Daten zum Prozessablauf ist eine Aufgabe des dem Prozess zugrundeliegenden Informationssystems. Da es sich nicht um eine Anforderung auf Prozessebene handelt, wird auf eine Aufnahme in den Anforderungskatalog verzichtet.

Klassifikation der Sicherheitsanforderungen. Im folgenden Absatz werden die zuvor erarbeiteten Sicherheitsanforderungen verschiedenen Eigenschaften zugeordnet, da sich daraus generelle Konsequenzen sowohl für ihre Überprüfung als auch ihre Entscheidbarkeit ergeben.

Die Anforderungen **(A1a)** bis **(A9)**, mit Ausnahme von **(A7b)**, können durch Eigenschaften über einzelne Prozessinstanzen ausgedrückt werden, d. h. es handelt sich um *Intra-Instanz-Sicherheitsanforderungen*. Für ihre Überprüfung müssen daher nur die Eigenschaften einzelner Instanzen betrachtet werden. Die in den vorangegangenen Schritten ermittelten Intra-Instanz-Sicherheitsanforderungen, mit Ausnahme von **(A2b)**, entsprechen sogenannten Safety-Eigenschaften. Eine Safety-Eigenschaft (*nothing bad will ever happen*) ist grundsätzlich beobachtbar und somit entscheidbar [54]. Das bedeutet konkret, es kann zu jedem Zeitpunkt eine Aussage darüber getroffen werden, ob eine Verletzung einer Safety-Eigenschaft bzw. der entsprechenden Anforderung vorliegt oder nicht. Bei Anforderung **(A2b)** handelt es sich um eine Liveness-Eigenschaft (*something good will eventually happen*) [17]. Sie ist nicht generell entscheidbar, da die

zukünftige Ausführung der Aktivität Y zwar vorgeschrieben ist, dies aber nicht zwingend in endlicher Zeit stattfinden muss. Sobald Y stattgefunden hat, steht zwar fest, dass die Anforderung erfüllt ist, aber zu keinem Zeitpunkt kann eine Aussage über das Vorliegen einer Verletzung getroffen werden. An dieser Stelle wird die Anforderung **(A2b)** daher in eine restriktivere, aber beobachtbare Anforderung überführt, indem sie um eine vorgeschriebene Zeitspanne erweitert wird:

> **(A2b*)** Wird Aktivität X ausgeführt, muss danach Aktivität Y innerhalb einer endlichen Zeitspanne Z ausgeführt werden.

Diese Anforderung ist entscheidbar. Anforderung **(A5)**, die ebenfalls eine zeitbeschränkte und somit beobachtbare Obligation repräsentiert, ist äquivalent zu Anforderung **(A2b*)**. Sie kann daher aus dem Katalog gestrichen werden.

Bei den Anforderungen **(A10)** und **(A11)** zur Informationsflusskontrolle sowie **(A7b)** handelt es sich um *Inter-Instanz-Sicherheitsanforderungen*. Da sie nicht nur einzelne Instanzen des Prozesses betreffen, sondern auf Eigenschaften über alle Instanzen aufbauen, können sie nicht durch eine einzelne Safety- oder Liveness-Eigenschaft beschrieben werden. Inter-Instanz-Anforderungen können aber durch sogenannte Hypereigenschaften (*hyperproperties*, Eigenschaften über Eigenschaften) ausgedrückt werden [29]. Die Überprüfung von Hypereigenschaften erfordert entweder den Vergleich zwischen den Eigenschaften einzelner Elemente von verschiedenen Instanzen oder einen Zwischenschritt, in welchem zunächst eine Repräsentation (z. B. ein Modell) der zugrundeliegenden Eigenschaften erstellt wird. Basierend auf einer solchen Repräsentation können anschließend weitere Eigenschaften festgelegt und auch entschieden werden (siehe auch [12]).

Tabelle 2.4 enthält eine vollständige Auflistung der Sicherheitsanforderungen und repräsentiert den Anforderungskatalog, der als Grundlage für die Fallstudien dieser Arbeit dient.

Tabelle 2.4.: Katalog von Sicherheitsanforderungen mit der Klasse ihrer zugehörigen Eigenschaft (S: Safety-Eigenschaft, H: Hypereigenschaft).

Name	Klasse	Beschreibung
(A1a)	S	Aktivität X darf nicht enthalten sein.
(A1b)	S	Aktivität X muss enthalten sein.
(A2a)	S	Wird Aktivität X ausgeführt, muss zuvor Aktivität W ausgeführt worden sein.
(A2b*)	S	Wird Aktivität X ausgeführt, muss danach Aktivität Y innerhalb einer endlichen Zeitspanne Z ausgeführt werden.
(A3)	S	Alle Prozessabläufe entsprechen vorgegebenen Reihenfolgen von Aktivitäten, die z. B. durch ein Prozessmodell definiert werden.
(A4)	S	Auf einen Entscheidungspunkt folgt Aktivität X falls Bedingung B wahr ist, Aktivität Y anderenfalls.
(A6a)	S	Aktivität X darf nur von einem autorisierten Teilnehmer T ausgeführt werden.
(A6b)	S	Alle Teilnehmer müssen die Berechtigungen der RBAC-Vorgaben einhalten.
(A7a)	S	Führt Teilnehmer $T1$ die Aktivität X aus, so muss die Aktivität Y in der *gleichen Instanz* des Prozesses von einem Teilnehmer $T2$ ausgeführt werden, so dass $T1 \neq T2$ gilt.
(A7b)	H	Führt Teilnehmer $T1$ die Aktivität X aus, so muss die Aktivität Y in einer *anderen Instanz* des Prozesses von einem Teilnehmer $T2$ ausgeführt werden, so dass $T1 \neq T2$ gilt.
(A8)	S	Aktivität X mit Zugriff auf Daten D darf nur von einem autorisierten Teilnehmer T ausgeführt werden.
(A9)	S	Alle Teilnehmer die an der Ausführung einer Prozessinstanz beteiligt sind, müssen zur gleichen Partei gehören.
(A10)	H	Prozessaktivitäten, die Eingabe/Ausgabe-Operationen realisieren, dürfen bei der Ausführung durch Teilnehmer verschiedener Parteien keine Daten als Eingabe verwenden, die von einer anderen Partei erzeugt, bzw. ausgegeben wurden.
(A11)	H	Teilnehmer am Prozess dürfen nicht ableiten können, welche Bedingungen über Daten die Prozessausführung beeinflussen, z. B. bei einer automatisch durchgeführten Berechnung auf Grundlage von Daten (Isolation).

3. Process Mining

Dieses Kapitel ist einer Beschreibung des Process Minings gewidmet. Im weitesten Sinne lässt sich das Process Mining als *Technik zur Gewinnung von Wissen aus Ereignisprotokollen von Informationssystemen* definieren. Ein Ereignisprotokoll ist demnach die wesentliche Grundlage und Mindesteingabe für alle Process Mining-Verfahren. Aufbau, Bestandteile und Anforderungen an ein Ereignisprotokoll, das für Process Mining eingesetzt werden soll, werden im nächsten Abschnitt 3.1 zusammen mit einer formalen Repräsentation beschrieben.

Die Verfahren des Process Minings können sowohl nach Typen als auch nach Perspektiven klassifiziert werden (Abschnitt 3.2). In dieser Arbeit werden die Typen Modell-Extraktion sowie Konformitätsprüfung behandelt und entsprechende Verfahren in den jeweiligen Abschnitten 3.3 und 3.4 vorgestellt.

3.1. Prozessorientiertes Ereignisprotokoll

Informationssysteme zeichnen während ihrer Ausführung eine Vielzahl an Daten auf. Beim Process Mining werden diese Daten aus einer prozessorientierten Perspektive betrachtet. Um eine solche Perspektive zu ermöglichen, müssen grundlegende Anforderungen an den Aufbau und die zu enthaltenden Informationen eines Ereignisprotokolls erfüllt werden [3]:

- Ein Ereignisprotokoll bezieht sich auf einen einzelnen Prozess
- Ein Prozess beinhaltet *Instanzen* (*cases*), die konkrete Fälle, d. h. Prozessabläufe beschreiben
- Eine Instanz setzt sich aus *Ereignissen* (*events*) zusammen
- Ereignisse können *Attribute* besitzen. Typische Beispiele für Attribute sind Aktivität, Teilnehmer und Zeitstempel.

In Tabelle 3.1 wird ein Beispiel für einen Auszug aus einem entsprechenden Ereignisprotokoll vorgestellt. Das Beispiel enthält zwei abgeschlossene Instanzen (1 und 2) eines Prozesses zur Bearbeitung von Anträgen zur Kostenerstattung bei einer Versicherung. Die Zeilen der Tabelle repräsentieren die einzelnen Ereignisse

Tabelle 3.1.: Beispiel für einen Auszug aus einem Ereignisprotokoll welches eine prozessorientierte Perspektive ermöglicht. Die Ereignisse sind nach a) Instanzen und b) Zeitstempeln sortiert.

Instanz	Aktivität	Teilnehmer	Betrag	Zeitstempel
1	*Antrag registrieren*	Maria	1000 €	2011-12-24, 11:10:21
1	*Antrag prüfen*	Max	-	2011-12-24, 11:15:21
1	*Antrag ablehnen*	Max	-	2011-12-24, 12:17:10
1	*Antrag abschließen*	Maria	-	2011-12-24, 12:47:11
2	*Antrag registrieren*	Maria	200 €	2011-12-24, 11:31:38
2	*Antrag prüfen*	Max	-	2011-12-24, 11:48:28
2	*Erstattung berechnen*	Felix	-	2011-12-24, 11:50:21
2	*Erstattung auszahlen*	Felix	-	2011-12-24, 12:14:56
2	*Antrag abschließen*	Maria	-	2011-12-24, 12:18:38

dieses Prozesses. Aus ihnen lassen sich die jeweiligen Vorgänge direkt ablesen. Die Spalten der Tabelle entsprechen dabei den vorhandenen Attributen der Ereignisse.

Die zwei Spalten Instanz und Aktivität beschreiben die minimal notwendigen Informationen. Sollen Abfolge oder kausale Zusammenhänge von Aktivitäten erkannt werden (wie z. B. bei der Modell-Extraktion), so muss entweder zusätzlich ein Zeitstempel vorhanden sein, nach welchem sich die Ereignisse ordnen lassen, oder die Ereignisse im Ereignisprotokoll müssen bereits in chronologischer Reihenfolge vorliegen. In Abhängigkeit vom einzusetzenden Verfahren werden auch weitere Informationen, z. B. Teilnehmer oder andere Datenfelder, benötigt oder berücksichtigt.

An dem Beispiel in Tabelle 3.1 ist der Umstand zu erkennen, dass Ereignisse auch unterschiedliche Attribute besitzen können. Hier enthalten die Ereignisse bezogen auf die Aktivität *Antrag registrieren* ein zusätzliches Attribut für den Betrag des eingehenden Antrags.

Basierend auf den bereits beschriebenen Elementen kann ein Ereignisprotokoll formal definiert werden. Nach van der Aalst [3] können zunächst die Begriffe Ereignis und Attribut wie folgt festgelegt werden:

Definition 4 (Ereignis, Attribut)**.** Sei $\mathscr{E}$ das Ereignisuniversum, d. h. die Menge aller möglichen Bezeichnungen für *Ereignisse*. Ereignisse können durch verschiedene *Attribute* charakterisiert werden. Sei AN die Menge von Attributnamen. Für jedes Ereignis $e \in \mathscr{E}$ und jeden Namen $n \in AN$ gilt: $\#_n(e)$ ist der Wert des Attributs n für Ereignis e. Besitzt Ereignis e kein Attribut namens n, so gilt: $\#_n(e) = \perp$ (Null-Wert).

Gemäß Definition 4 lassen sich z. B. die Attribute des Ereignisses e_2 in der zweiten Zeile des Beispiels in Tabelle 3.1 auf folgende Weise ausdrücken: $\#_{Aktivitaet}(e_2) = \textit{Antrag prüfen}$, $\#_{Ressource}(e_2) = Max$, $\#_{Betrag}(e_2) = \perp$, $\#_{Zeitstempel}(e_2) = \textit{2011-12-24, 11:15:21}$.

Darauf aufbauend kann die Definition der Begriffe Instanz, Trace und Ereignisprotokoll angegeben werden.

Definition 5 (Instanz, Trace, Ereignisprotokoll)**.** Sei $\mathscr{I}$ das Instanzuniversum, d. h. die Menge aller möglichen Bezeichnungen ($\#_{Instanz}$) für *Instanzen*. Wie Ereignisse haben auch Instanzen Attribute, die analog definiert werden: Für jede Instanz $i \in \mathscr{I}$ und jeden Namen $n \in AN$ gilt: $\#_n(i)$ ist der Wert des Attributs n für die Instanz i. Für jede Instanz $i \in \mathscr{I}$ lässt sich ein zugehöriger *Trace* bilden: ein Trace ist eine endliche Sequenz von Ereignissen $\sigma = \langle e_1, e_2, \ldots, e_n \rangle \in E^*$ wobei alle Ereignisse dieses Trace den gleichen Instanzbezeichner haben, bzw. $\#_{Instanz}(e_x) = \#_{Instanz}(e_y)$ für alle $1 \leq x < y \leq n$ und darüber hinaus maximal einmal in der Sequenz auftreten. Ein *Ereignisprotokoll* ist eine Menge von Instanzen $L \subseteq \mathscr{I}$, für die gilt, dass jedes Ereignis eindeutig ist und maximal einmal im gesamten Protokoll auftaucht.

Somit besteht ein Ereignisprotokoll aus Instanzen und Instanzen wiederum bestehen aus Ereignissen. Die Ereignisse einer Instanz werden in Form eines Traces, d.h. einer Sequenz von eindeutigen Ereignissen repräsentiert. Darüber hinaus können Instanzen wie auch Ereignisse Attribute besitzen. $L = \{1, 2\}$ entspricht dem Ereignisprotokoll in Tabelle 3.1 mit den zwei Instanzen 1 und 2. Der Trace zur Instanz 1 wird durch $\sigma(1) = \langle e_1, e_2, e_3, e_4 \rangle$ angegeben, wobei e_1–e_4 als eindeutige Bezeichner aus $\mathscr{E}$ für die Ereignisse in den Zeilen 1–4 der Tabelle fungieren. Das vorgestellte Beispiel enthält keine Instanzattribute. Denkbar wäre es, z. B. für eine Instanz einen verantwortlichen Sachbearbeiter festzulegen: $\#_{Sachbearbeiter}(1) = Helga$, so dass Helga für die Koordination der Instanz 1 zuständig ist.

Durch die vorgestellte Formalisierung von Ereignisprotokollen werden die Anforderungen präzise festgelegt, ohne ein konkretes Format (wie z. B. MXML oder XES, siehe Abschnitt 4.2.1) zu definieren. Basierend auf dieser formalen Repräsentation liefert ein prozessorientiertes Ereignisprotokoll die Ausgangsgrundlage für Process Mining-Verfahren und kann von diesen ausgewertet werden, um Wissen über den zugrundeliegenden Prozess zu gewinnen.

Dabei spielt auch der sogenannte Reifegrad des Ereignisprotokolls eine Rolle [47]. Zur Feststellung des Reifegrads werden die im Ereignisprotokoll enthaltenen Ereignisse als fundamentale Informationsträger aufgefasst und die Qualität der Ereignisdaten nach einer Reihe von Kriterien beurteilt. Ereignisse sind *belastbar*, wenn sicher ist, dass sie tatsächlich stattgefunden haben und die Attributwerte

Tabelle 3.2.: Reifegrade von Ereignisprotokollen. Quelle: Process Mining Manifesto [47]

Grad	Beschreibung	Beispiele
★★★★★	Automatische, systematische, zuverlässige und sichere Aufzeichnung. Das Ereignisprotokoll ist vollständig, die aufgezeichneten Ereignisse sind belastbar und besitzen eine wohldefinierte Semantik. Datenschutz und Sicherheitsaspekte werden angemessen berücksichtigt.	Semantisch annotierte Ereignisprotokolle von BPM-Systemen
★★★★	Automatische, systematische und zuverlässige Aufzeichnung. Das Ereignisprotokoll ist vollständig, die aufgezeichneten Ereignisse sind belastbar. Konzepte wie Prozessinstanzen (Fälle) und Aktivitäten werden explizit unterstützt.	Ereignisprotokolle klassischer WfMS oder BPM-Systeme
★★★	Automatische Aufzeichnung ohne systematischen Ansatz, es gibt jedoch im Gegensatz zu Ereignisprotokollen des Grades ★★ ein gewisses Maß an Sicherheit, dass die aufgezeichneten Ereignisse der Realität entsprechen. Die aufgezeichneten Ereignisse sind daher belastbar, aber das Ereignisprotokoll ist nicht zwangsläufig vollständig.	Datenbanktabellen aus ERP- oder Ereignisprotokolle von CRM-Systemen
★★	Automatische, jedoch keine systematische Aufzeichnung bei der es möglich ist, das Informationssystem zu umgehen. Das Ereignisprotokoll ist daher nicht zwangsläufig vollständig und Ereignisse nicht zwangsläufig belastbar.	Ereignisprotokolle von Dokumentverwaltungen oder Produktionssystemen, Arbeitsübersichten
★	Keine automatische Aufzeichnung, oft händische Erstellung. Das Ereignisprotokoll ist weder vollständig, noch sind die aufgezeichneten Ereignisse belastbar.	Vermerke in Dokumenten, die innerhalb einer Organisation zirkulieren, schriftliche Krankenakten

korrekt (d. h. authentisch) sind. Ein Ereignisprotokoll ist *vollständig*, wenn keine Ereignisse in der Aufzeichnung fehlen. Jedes Ereignis sollte darüber hinaus eine wohldefinierte *Semantik* besitzen, zudem müssen im Rahmen der Aufzeichnung *Datenschutz und Sicherheitsaspekte* berücksichtigt werden. Dies schließt die Aufklärung von Teilnehmern über die Aufzeichung und den Verwendungszweck der Ereignisdaten ein. Nach Tabelle 3.2 werden fünf verschiedene Reifegrade von ★ (niedrigster Grad) bis ★ ★ ★ ★ ★ (höchster Grad) für Ereignisprotokolle definiert.

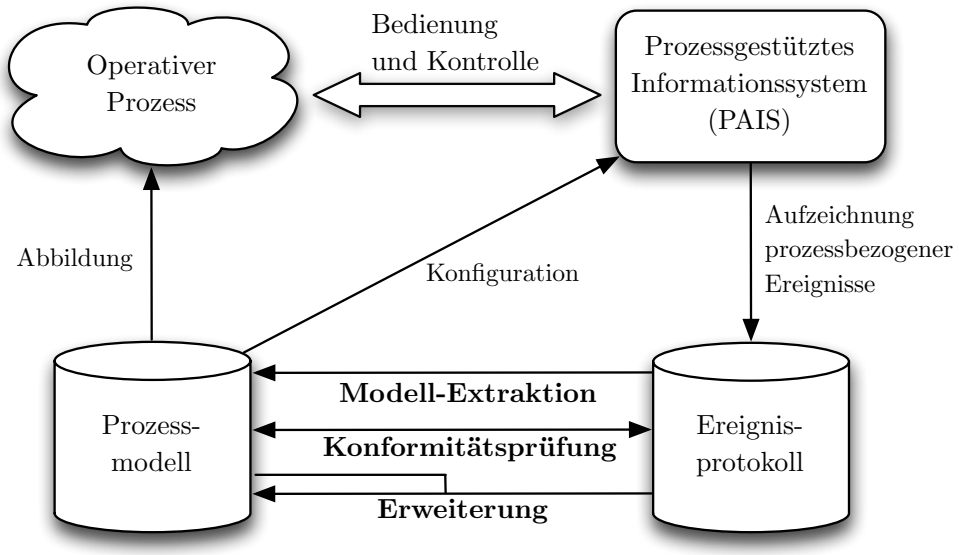

Abbildung 3.1.: Typen von Process Mining-Verfahren. Die Modell-Extraktion benötigt ein Ereignisprotokoll als Eingabe, die Konformitätsprüfung und Erweiterung darüber hinaus ein Prozessmodell. Modifiziert nach van Dongen [33, S. 8]; van der Aalst [3, S. 9].

3.2. Typen und Perspektiven

Auf der Grundlage von Ereignisprotokollen können in erster Linie drei wesentliche und vielzitierte Typen von Process Mining-Verfahren zur Anwendung kommen. Abbildung 3.1 zeigt eine Übersicht dieser Typen im Kontext ihrer Ein- und Ausgaben sowie PAIS und operativen Prozessen. Operative Prozesse dienen der eigentlichen Leistungserstellung [70, S. 55]. Sie kennzeichnen die tatsächliche Ausführung von Prozessaktivitäten (Ereignissen) unabhängig davon, ob diese manueller oder automatisierter Natur sind. Es findet eine wechselseitige Steuerung und Kontrolle zwischen operativen Prozessen und PAIS statt. Mit Hilfe von PAIS werden laufend Aufzeichnungen über Prozessaktivitäten in Ereignisprotokollen gespeichert, so dass eine Dokumentation des tatsächlich stattfindenden (konkreten) Prozessablaufs erfolgt. Diesem steht ein (abstraktes) Prozessmodell gegenüber. Das Process Mining setzt nun (a) Prozessmodelle und (b) Ergeignisprotokolle in Beziehung zueinander. Daraus ergeben sich drei Typen von Zielen für Process Mining-Verfahren:

1) **Modell-Extraktion** *(Process Discovery)*: Verfahren zur Modell-Extraktion verwenden als Eingabe ein Ereignisprotokoll und geben ein Prozessmodell aus. Zur Konstruktion des Prozessmodells werden ausser dem Ereignisprotokoll keine weiteren *a priori* Informationen verwendet. Beispielsweise kann mit dem α-Algorithmus [9] ein Petri-Netz-basiertes Prozessmodell konstruiert werden (siehe auch Abschnitt 3.3). Enthalten die Ereignisse des Ereignisprotokolls entsprechende Attribute ($\#_{Ressource}$), können Verfahren zur Konstruktion von Modellen sozialer Netzwerke [71] angewendet werden. Dadurch lassen sich Beziehungen zwischen Ressourcen (Teilnehmern) des Prozesses repräsentieren. Extrahierte Modelle können als Basis für weiterführende Analyseverfahren dienen.

2) **Konformitätsprüfung** *(Conformance Checking)*: Eingaben sind hier i) ein bestehendes Prozessmodell sowie ii) ein auf den gleichen Prozess bezogenes Ereignisprotokoll. Verfahren zur Konformitätsprüfung vergleichen diese beiden Eingaben und geben eine Diagnose bezüglich ihrer Übereinstimmung aus. Diese zeigt auf, inwiefern die im Ereignisprotokoll dokumentierte Realität tatsächlich mit dem Modell übereinstimmt und vice versa. Mit der Petri-Netz-basierten Konformitätsprüfung [65] können z. B. Traces eines Ereignisprotokolls gemäß ihrer Anwendbarkeit auf ein gegebenes Petri-Netz in konforme und nicht-konforme Traces klassifiziert werden (siehe auch Abschnitt 3.4).

3) **Erweiterung** *(Enhancement)*: Verfahren zur Erweiterung benötigen als Eingabe ebenfalls (i) ein bestehendes Prozessmodell und (ii) ein auf den gleichen Prozess bezogenes Ereignisprotokoll. Das Ziel dieser Verfahren ist es, das gegebene Prozessmodell auf Basis der Informationen aus dem Ereignisprotokoll zu erweitern und zu verbessern, um ein aussagekräftigeres oder optimiertes Prozessmodell zu erhalten. So können z. B. Prozessmodelle, die den Kontrollfluss repräsentieren, um Entscheidungsregeln erweitert werden, die für XOR-Splits angeben, wie auf der Grundlage von Attributwerten ein entsprechender Pfad ausgewählt wird [66].

Orthogonal zu den soeben vorgestellten drei Typen lassen sich Process Mining-Verfahren auch nach Perspektiven klassifizieren [3]:

- **Kontrollfluss-Perspektive**: Der Fokus liegt auf dem Kontrollfluss, z. B. konstruiert ein Verfahren zur Modell-Extraktion ein Prozessmodell, das Abfolge und kausale Zusammenhänge von Aktivitäten im Prozess abbildet. Ziel ist es, eine präzise Charakterisierung aller möglichen Pfade zu erhalten.
- **Organisatorische Perspektive**: Enthält das Ereignisprotokoll Informationen zu verwendeten Ressourcen wie Teilnehmer, Rollen, Systeme oder

Abteilungen, können Process Mining-Verfahren diese nutzen, um z. B. vorhandene Beziehungen darzustellen (soziale Netzwerke) oder Rollenmodelle abzuleiten.

- **Fallperspektive**: Hier stehen Eigenschaften einzelner Fälle (Instanzen) im Vordergrund. Ein Fall kann beispielsweise durch seine Teilnehmer oder Werte verschiedener Datenfelder charakterisiert werden.
- **Zeitperspektive**: Wenn im Ereignisprotokoll Zeitstempel (oder die Dauer) zu einzelnen Ereignissen festgehalten werden, kann die gezielte Analyse dazu beitragen, z. B. Engpässe zu identifizieren, die Verwendung von Ressourcen zu beobachten oder die verbleibende Zeit des Prozessablaufs in einem Fall vorherzusagen.

Die hier vorgestellten Perspektiven sind nicht erschöpfend und können auch teilweise überlappen. Dennoch ermöglichen sie eine grobe Charakterisierung der Aspekte, die bei der Analyse von Process Mining-Verfahren im Vordergrund stehen können.

3.3. Modell-Extraktion

Verfahren zur Modell-Extraktion extrahieren aus einem Ereignisprotokoll ein Modell. Diese Modelle können, wie im letzten Abschnitt beschrieben, unterschiedliche Perspektiven auf den Prozess ermöglichen. Abschnitt 3.3.1 erläutert das Ziel von Verfahren zur Extraktion von Kontrollfluss-Modellen und gibt einen Überblick über entsprechende Verfahren. In Abschnitt 3.3.2 werden Verfahren betrachtet, mit welchen Modelle über die Organisation eines Prozesses, wie z. B. soziale Netzwerke, gewonnen werden können.

3.3.1. Kontrollfluss

Das Ziel von Verfahren zur Modell-Extraktion ist es, ein Prozessmodell zu konstruieren, das die Abfolge und kausale Zusammenhänge von Prozessaktivitäten präzise abbildet. Was in diesem Zusammenhang der Begriff *präzise* bedeutet, soll zunächst an einem Beispiel erläutert werden. In Abb. 3.2 werden verschiedene Modelle für einen einfachen Prozess zur Abwicklung von Abrechnungen dargestellt. Für den Prozessablauf ist vorgesehen, dass zuerst die Abrechnung gestartet und danach eine Rechnung versendet wird. Anschließend wird, falls nötig, in regelmäßigen Abständen eine Zahlungserinnerung versendet, bis der Zahlungseingang registriert wird, danach kann die Abrechnung geschlossen werden. Ein

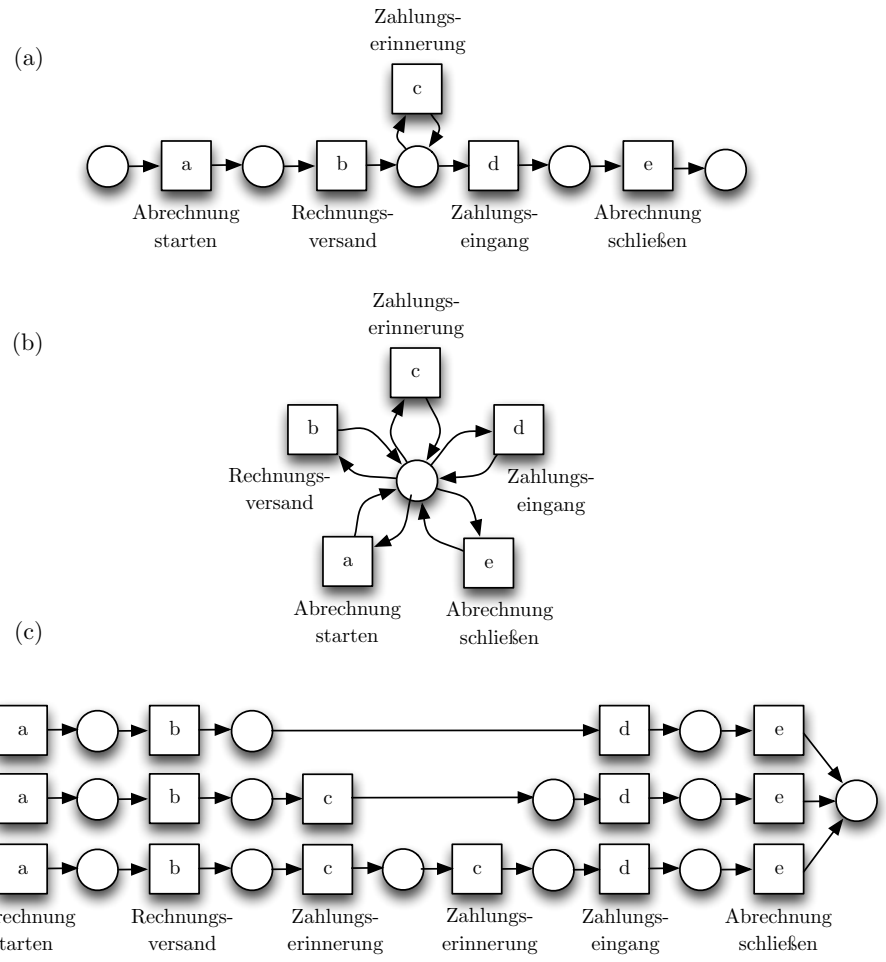

Abbildung 3.2.: Verschiedene Prozessmodelle für den gleichen Prozess: (a) präzises Modell, (b) verallgemeinerndes Modell, (c) spezifisches oder explizites Modell. Modifiziert nach Alves de Medeiros [56, S. 9]

Ereignisprotokoll bestehend aus drei Traces mit den folgenden Aktivitätssequenzen $\sigma(1) = \langle a, b, c, e \rangle$, $\sigma(2) = \langle a, b, c, d, e \rangle$ und $\sigma(3) = \langle a, b, c, d, d, e \rangle$ passt gleichermaßen zu allen drei in der Abbildung gezeigten Prozessmodellen. Es ist trivial, auf Grundlage eines jeden Ereignisprotokolls ein Prozessmodell zu entwerfen, zu dem alle Traces passen, wie im Folgenden gezeigt wird. Das verallgemeinernde Modell (b), auch Flower-Modell genannt, lässt alle möglichen Abfolgen der Prozessaktivitäten zu, also auch die drei Traces des vorgestellten

Ereignisprotokolls. Da darüber hinaus aber auch alle anderen möglichen Abfolgen der Prozessaktivitäten zu dem Modell passen, besitzt es keinerlei Informationsgehalt über den Prozessablauf und ist ungeeignet, das durch das Ereignisprotokoll dokumentierte Verhalten wiederzuspiegeln. Das andere Extrem ist das spezifische oder auch explizite Modell (c), welches jeden einzelnen Trace des Prozesses als eigenen Pfad ausdrückt. Es spiegelt exakt das durch das Ereignisprotokoll dokumentierte Verhalten wieder, ohne von diesem zu abstrahieren. So entspricht der Informationsgehalt dieses Modells lediglich einem direkten Blick auf die Aktivitätssequenzen der Traces des Ereignisprotokolls selbst. Darüber hinaus passen zu dem Modell keine anderen Abfolgen von Prozessaktivitäten, als diejenigen die im Ereignisprotokoll enthalten sind. So kann z. B. die Aktivität *Zahlungserinnerung* nicht öfter als zweimal hintereinander ausgeführt werden, was nicht dem vorgesehenen Prozessablauf entspricht.

Eine Balance zwischen diesen beiden Extremen kann erst durch die nichttriviale Extraktion des präzisen Modells (a) erreicht werden. Tatsächlich drückt das Modell (a) genau den vorgesehenen Prozessablauf aus. Alle drei Aktivitätssequenzen des Ereignisprotokolls passen zu dem Modell, ohne dass dieses zu spezifisch aufgebaut ist oder zu sehr verallgemeinert. Im Folgenden wird ein Überblick über eine Reihe von Verfahren zur Modell-Extraktion gegeben, deren gemeinsames Ziel die Konstruktion eines solchen präzisen Modells ist.

Zu den ersten Verfahren der Modell-Extraktion wird der Ansatz von Cook u. Wolf [30] gezählt, mit dem Prozessmodelle aus Ereignisprotokollen im Kontext des Software Engineering gewonnen werden können. Die Anwendung der Modell-Extraktion in einem Workflow-Kontext wurde erstmals von Agrawal et al. [16] durchgeführt. Weitere Beiträge stammen von Datta [31], Golani u. Pinter [40], Greco et al. [41], Herbst [46], Schimm [69]. Verfahren zur Modell-Extraktion basieren vielmals auf grundlegend unterschiedlichen Ansätzen und Methoden, um diverse Schwierigkeiten zu adressieren, die sich bei der Konstruktion von Prozessmodellen auf Grundlage von Ereignisprotokollen auftun, z. B. die Erkennung von strukturellen Mustern wie Nebenläufigkeit und Schleifen (cf. Abschnitt 2.2) oder die Behandlung von vom vorgesehenen Prozessablauf abweichenden Trace (*Noise*) im Ereignisprotokoll (cf. Abschnitt 4.2.2). Aufgrund der großen Anzahl und der Komplexität unterschiedlicher Verfahren wird im Folgenden ein Einblick in die Vorgehensweise einer kleinen Auswahl aktueller Ansätze algorithmischer Natur gewährt, die auch in [3, 36, 44] beschrieben werden.

Der **α-Algorithmus** von van der Aalst et al. [9] unterscheidet sich von vielen Vorgänger-Verfahren darin, dass er von Nebenläufigkeit in Prozessen ausgeht und auf Petri-Netz-Prozessmodellen basiert, die gut geeignet sind, nebenläufige Prozesse zu repräsentieren. Die Konstruktion eines Petri-Netz Prozessmodells

basiert auf dem Konzept von Ordnungsbeziehungen zwischen Aktivitäten im Ereignisprotokoll gemäß Definition 6 [3].

Definition 6 (Log-basierte Ordnungsbeziehungen)**.** Sei L ein Ereignisprotokoll über der Menge von Aktivitäten $a, b, c, \cdots \in \mathscr{A}$. Für zwei Aktivitäten $a, b \in \mathscr{A}$ gilt $a >_L b$ gdw. es einen Trace $\sigma = \langle t_1, t_2, \ldots, t_n \rangle$ gibt, so dass $\sigma \in L$ und $t_i = a$ und $t_{i+1} = b$, wobei $i \in \{1, \ldots, n-1\}$. Somit drückt $a >_L b$ aus, dass es einen Trace im Ereignisprotokoll gibt, in welchem die Aktivität b direkt auf die Aktivität a folgt (b *follows* a). Basierend auf der grundlegenden Ordnungsbeziehung $>_L$ kann jeweils eine von drei weiteren Ordnungsbeziehungen für jedes Paar aus Aktivitäten $a, b \in \mathscr{A}$ abgeleitet werden:

- $a \rightarrow_L b$ gdw. $a >_L b$ und $b \not>_L a$ (*causal relation*)
- $a \#_L b$ gdw. $a \not>_L b$ und $b \not>_L a$ (*unrelated*)
- $a \parallel_L b$ gdw. $a >_L b$ und $b >_L a$ (*parallel relation*)

Ist ein Ereignisprotokoll vollständig in Bezug auf die grundlegende Ordnungsbeziehung $>_L$ ($follows$) und darüber hinaus frei von Noise, funktioniert dieser Ansatz korrekt für eine eingeschränkte Teilklasse von Petri-Netzen, den sogenannten strukturierten Workflow-Netzen (siehe [9] für den Beweis). Vollständigkeit bedeutet hier, dass falls im vorgesehenen Prozessablauf eine grundlegende Ordnungsbeziehung $a >_L b$ zwischen zwei Aktivitäten a und b besteht, diese mindestens einmal im Ereignisprotokoll auftreten muss.

Da für die vorgestellten binären Ordnungsbeziehungen zwischen Aktivitäten auf denen der Algorithmus basiert, die booleschen Werte wahr oder falsch gelten, wird die Häufigkeit dieser Beziehungen nicht einbezogen. Das bedeutet, dass eine einzige Abweichung vom vorgesehenen Prozessablauf im Ereignisprotokoll die Ordnungsbeziehungen und somit auch das resultierende Prozessmodell verändert. Daher ist der α-Algorithmus im Hinblick auf Noise nicht robust [44].

Aus diesem Gesichtspunkt können die Arbeiten zum **heuristischen Mining** von Weijters u. van der Aalst [77] als Erweiterung des α-Algorithmus verstanden werden, da hier der Fokus darauf liegt, den ursprünglichen Ansatz unempfindlicher gegen Noise zu machen. Dabei bildet die grundlegende Ordnungsbeziehung $>_L$ des α-Algorithmus nicht mehr auf boolesche Werte bzw. wahr oder falsch ab, sondern repräsentiert die Häufigkeit des Auftretens der Beziehungen im Ereignisprotokoll. Auf dieser Grundlage werden dann Heuristiken eingesetzt, um die weiteren Ordnungsbeziehungen abzuleiten. Als Beispiel sollen in einem Ereignisprotokoll sowohl die Beziehungen $a >_L b$ als auch $b >_L a$ auftreten. Die Häufigkeit der Beziehung $a >_L b$ sei sehr viel höher, als die der Beziehung $b >_L a$. Der α-Algorithmus wird darauf basierend für die Aktivitäten a und b

die Ordnungsbeziehung $a||_L b$ wählen, wohingegen der heuristische Ansatz die selten auftretende Beziehung $b >_L a$ ignoriert und stattdessen für a und b die Ordnungsbeziehung $a \rightarrow_L b$ wählt.

Der Ansatz des heuristischen Mining ist zu einer der Standardanwendungen für die Durchführung der Modell-Extraktion in Fallstudien mit Real-Life-Daten geworden. Seine Unempfindlichkeit gegenüber Noise und die Fähigkeit zur Abstraktion, d.h. die Annhäherung an ein präzises Prozessmodell, machen ihn zu einem probaten Mittel für die praktische Anwendung [44].

Eine weitere Optimierung des Ergebnisses von Verfahren zur Modell-Extraktion kann durch die Anwendung des sogenannten **genetischen Mining** von Alves de Medeiros [56] erreicht werden. Diesem Ansatz liegt die Idee zugrunde, bei der Konstruktion von Prozessmodellen den Evolutionsprozess der Natur zu imitieren. Dabei werden zunächst verschiedene, zufällig erstellte Prozessmodelle als Kandidaten für eine mögliche Lösung gewählt und im weiteren als initiale Population von Individuen verwendet. Durch die Anwendung der genetischen Operatoren *Crossover* (Kombination zweier Individuen) und *Mutation* (Modifikation eines Individuums) wird die Zusammensetzung der Population in iterativen Schritten verändert. Dabei wählt ein Mechanismus zur Selektion auf Grundlage eines Fitness-Kriteriums die am besten zum Ereignisprotokoll passenden Individuen der Population als Eingabe für die nächsten Generationen aus. Somit wird bei jeder Iteration des Algorithmus die Fitness der Population erhöht und nach Erreichen eines definierten Stop-Kriteriums das Individuum, bzw. Prozessmodell mit der höchsten Fitness zurückgegeben.

Da die Operatoren des genetischen Algorithmus nicht nur Beziehungen zwischen Aktivitäten modifizieren, sondern auch Aktivitäten entfernen oder duplizieren, können mit dem Ansatz oft Prozessmodelle gewonnen werden, die sich besonders durch gute Lesbarkeit und Übersichtlichkeit auszeichnen [44].

3.3.2. Organisation

Die Verfahren von Song u. van der Aalst [71] zur Modell-Extraktion mit Fokus auf die organisatorische Perspektive extrahieren Modelle, die die Beziehungen zwischen Teilnehmern an einem Prozess in Form sozialer Netzwerke darstellen. Die Beziehungen zwischen Teilnehmern ergeben sich dabei auf Grundlage jeweils einer von folgenden Metriken:

- **Übergabe von Arbeit** (*handover of work*): In einem Fall (bzw. in einer Prozessinstanz) findet die Übergabe von Arbeit von einem Teilnehmer i zu einem Teilnehmer j dann statt, wenn es zwei aufeinanderfolgende Aktivitäten gibt, so dass die erste von i und die zweite von j ausgeführt

wird. Es ist dabei möglich, nicht nur direktes sondern auch indirektes Aufeinanderfolgen zu betrachten, indem ein Faktor β eingeführt wird, der die Entfernung beschreibt. Befinden sich drei Aktivitäten zwischen einer Aktivität, die von i und einer Aktivität die von j ausgeführt wird, so ist $\beta = 3$.

- **Zusammenarbeit** (*working together*): Diese Metrik ignoriert kausale Abhängigkeiten und zählt, wie oft zwei Teilnehmer Aktivitäten in einer gemeinsamen Instanz ausführen. So besitzen zwei Teilnehmer i und j, die häufig gemeinsame Fälle bearbeiten, eine stärkere Beziehung als zwei Teilnehmer x und y, die selten zusammenarbeiten.
- **Ähnliche Aufgaben** (*similar task*): Diese Metrik betrachtet die Aktivitäten, die von einzelnen Teilnehmer ausgeführt werden. Sie beruht auf der Annahme, dass zwei Teilnehmer i und j, die ähnliche Aktivitäten ausführen, eine stärkere Beziehung besitzen, als zwei Teilnehmer x und y, die völlig unterschiedliche Aktivitäten ausführen.
- **Reassignment**: Ereignisse, bei denen explizit eine Aktivität von einem Teilnehmer i zu einem Teilnehmer j zugewiesen wird, können Hinweise auf eine Beziehungshierarchie liefern. Beispielsweise kann beobachtet werden, dass i häufig Aktivitäten zu j delegiert, aber nicht umgekehrt. Dies kann als Zeichen dafür interpretiert werden, dass i in der Hierarchie über j steht.

Die Knoten im Graph eines sozialen Netzwerks repräsentieren die Teilnehmer am Prozess. Die Kanten zwischen ihnen werden auf Grundlage der o.g. Metriken eingesetzt. Dabei ist es möglich, zentrale Elemente visuell besonders hervorheben zu lassen. Das können z. B. Kanten sein, die eine stärkere Beziehung zwischen zwei Teilnehmern repräsentieren, oder Knoten, die für Teilnehmer mit besonders vielen Beziehungen stehen.

Im Zusammenhang mit der organisatorischen Perspektive gibt es weitere Verfahren, z. B. solche die ein hierarchisches Rollenmodell aus dem Ereignisprotokoll extrahieren. Werden auf Basis des Ereignisprotokolls für jeden Teilnehmer die Teilmengen an Aktivitäten bestimmt, die er ausführt, so lassen identische Teilmengen auf gleiche Rollen schließen. Ein hierarchisches Rollenmodell drückt aus, welche Teilnehmer welche Aktivitäten ausführen, wobei übereinstimmende Teilmengen (also Rollen) gruppiert werden. Beginnend unten mit den Teilnehmern deren Teilmengen die meisten Elemente besitzen (die also die meisten Aktivitäten ausführen), werden nach oben hin weitere Teilmengen eingeführt und durch gerichtete Pfeile verbunden. Ein Pfeil von einer Teilmenge S_A für einen Teilnehmer T_1 zu einer Teilmenge S_B für Teilnehmer T_2, T_3 drückt dabei aus,

dass Teilnehmer T_1 die Aktivitäten in Teilmenge S_A, aber auch die Aktivitäten in Teilmenge S_B ausführt. Die Teilnehmer T_2, T_3 dahingegen führen nur die Aktivitäten in Teilmenge S_B aus.

3.4. Konformitätsprüfung

Process Mining-Verfahren zur Konformitätsprüfung vergleichen ein bestehendes Prozessmodell mit einem Ereignisprotokoll. Hierfür können unterschiedliche Modelle als Grundlage verwendet werden. In den folgenden Abschnitten werden die Petri-Netz-basierte Konformitätsprüfung (3.4.1) sowie die Log-basierte Verikfikation (3.4.2) vorgestellt.

3.4.1. Petri-Netz-basierte Konformitätsprüfung

Bei der Petri-Netz-basierten Konformitätsprüfung dient ein Petri-Netz als Grundlage für den Vergleich mit der im Ereignisprotokoll dokumentierten Realität. Die dabei eingesetzte Technik zum Abspielen eines Ereignisprotokolls auf dem gegebenen Petri-Netz wird Replay (auch *Token Replay*) genannt. Während des Replays wird für jeden Trace eine Marke in der initialen Stelle des Netzes positioniert und nacheinander diejenigen Transitionen gefeuert, die den jeweiligen Aktivitäten der Ereignisse im Trace entsprechen. Ein naiver Ansatz ist es, alle Traces eines Ereignisprotokolls auf dem gegebenen Petri-Netz abzuspielen und dabei zu prüfen, ob sich ein Trace abspielen lässt, oder nicht [3]. Darauf basierend kann die sogenannte einfache Fitness f_{simple} berechnet werden[1].

Definition 7 (Einfache Fitness)**.** Gegeben seien ein Petri-Netz P und ein Ereignisprotokoll L mit $|L|$ Traces. Die Menge $T_{OK} \subseteq L$ entspricht der Menge Traces aus dem Ereignisprotokoll, die auf dem Petri-Netz vollständig abgespielt werden können. Die Fitness-Metrik $f_{simple}(P, L)$ wird dann wie folgt definiert:

$$f_{simple}(P, L) = \frac{|T_{OK}|}{|L|}$$

Allerdings lässt die Fitness f_{simple} auf Basis der Klassifikation von Traces in die Kategorien „abspielbar“ und „nicht abspielbar“ keine Unterscheidung zwischen mehr oder weniger abspielbaren Traces zu. Ein Ereignisprotokoll L_1 zu dem in Abb. 3.2 gezeigten Prozessmodell (a) (hier P_a), dessen Traces jeweils die erste Aktivität *Abrechnung starten* fehlt, besitzt die gleiche Fitness

[1] Die Fitness f_{simple} wird auch von Bezerra et al. [24] als *Fitness Model Test Function* und von Pérez-Castillo et al. [63] als *Recall* definiert.

wie ein Ereignisprotokoll L_2, dessen Traces keine einzige Aktivität aus dem Prozessmodell P_a wiedergeben: $f_{simple}(P_a, E_1) = f_{simple}(P_a, E_2) = 0$.

Das Verfahren von Rozinat u. van der Aalst [65] bestimmt für die Messung der Übereinstimmung zwischen Modell und Ereignisprotokoll daher die sogenannte Marken-basierte Fitness f_{token}. Dabei wird während des Replays gezählt (i) wieviele Marken produziert und (ii) wieviele Marken konsumiert werden, aber auch (iii) wieviele Marken fehlen und künstlich produziert werden müssen, weil die Transition, welche dem nächsten abzuspielenden Ereignis im Trace entspricht, nicht aktiviert ist (fehlerhafte Prozessausführung), und (iv) wieviele Marken nach dem Replay aller Ereignisse aus dem Trace im Netz verbleiben (Prozess nicht ordentlich beendet). Basierend auf diesen Zahlen ist die Berechnung der Marken-basierten Fitness f_{token} für ein Ereignisprotokoll und ein Petri-Netz möglich. Zunächst soll die Berechnung der Fitness $f_{token}(\sigma, PN)$ für einen einzelnen Trace σ auf einem Petri-Netz PN vorgestellt werden:

Definition 8 (Marken-basierte Fitness bezogen auf einen Trace)**.** Gegeben seien ein Trace σ und ein markiertes Petri-Netz PN (siehe Definition 3). Sei p die Zahl produzierter Marken, k die Zahl konsumierter Marken, f die Zahl fehlender Marken und v die Zahl verbleibender Marken während des Replays des Trace σ auf dem gegebenen Petri-Netz PN. Die Fitness-Metrik $f_{token}(\sigma, PN)$ wird wie folgt definiert:

$$f_{token}(\sigma, PN) = \frac{1}{2}\left(1 - \frac{f}{k}\right) + \frac{1}{2}\left(1 - \frac{v}{p}\right)$$

Diese Formel berechnet zu gleichen Teilen eine Bestrafung für den Fall, dass Marken, die konsumiert wurden, während des Replays fehlen $(1 - \frac{f}{k})$ und den Fall, dass Marken übrigbleiben, die während des Replays produziert wurden $(1 - \frac{v}{p})$. Auf dieser Grundlage kann die Fitness auch über alle Traces eines Ereignisprotokolls auf einem Petri-Netz berechnet werden, wobei die Werte für alle Traces entsprechend summiert werden.

Definition 9 (Marken-basierte Fitness bezogen auf ein Ereignisprotokoll)**.** Gegeben seien ein Ereignisprotokoll L und ein markiertes Petri-Netz PN. Sei U die Menge unterschiedlicher Traces[2] in L. Für jeden Trace $\sigma \in U$ ist $U(\sigma)$ die Zahl der Instanzen, die auf diesen Trace reduziert wurden, p_σ die Zahl produzierter Marken, k_σ die Zahl konsumierter Marken, f_σ die Zahl fehlender Marken und v_σ die Zahl verbleibender Marken während des Replays des Trace σ auf dem

[2]Traces werden hier als gleich betrachtet, wenn die Aktivitäten ihrer Ereignisse übereinstimmen, d. h. wenn für zwei Traces $\sigma_1 = \langle e1_1, ..., e1_n\rangle, \sigma_2 = \langle e2_1, ..., e2_n\rangle$ und für alle $1 \leq i \leq n$ gilt: $\#_{Aktivitaet}(e1_i) = \#_{Aktivitaet}(e2_i)$.

gegebenen Petri-Netz PN. Die Fitness-Metrik $f_{token}(L, PN)$ wird wie folgt definiert:

$$f_{token}(L, PN) = \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in U} U(\sigma) \cdot f_\sigma}{\sum_{\sigma \in U} U(\sigma) \cdot k_\sigma}\right) + \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in U} U(\sigma) \cdot v_\sigma}{\sum_{\sigma \in U} U(\sigma) \cdot p_\sigma}\right)$$

Für den Wert von f_{token} nach Definition 8 und 9 gilt stets $0 \leq f_{token} \leq 1$.

Auf Grundlage der einfachen Fitness f_{simple} (Definition 7) oder der Markenbasierten Fitness f_{token} (Definition 8) kann eine Klassifikation von Traces ihrer Konformität nach vorgenommen werden. Dabei wird die Menge aller Traces T_{ALL} des Ereignisprotokolls in eine Menge konformer Traces T_{OK} ($f = 1$, bzw. erfolgreiches Replay) und eine Menge nicht-konformer Traces T_{NOK} ($f \leq 1$, bzw. nicht erfolgreiches Replay) aufgeteilt.

Weiterhin geben Rozinat u. van der Aalst noch Metriken für die strukturelle Angemessenheit (*structural appropriateness*) und die behaviorale Angemessenheit (*behavioral appropriateness*) an. Da diese für die vorliegende Arbeit nicht relevant sind, wird an dieser Stelle auf eine Beschreibung verzichtet und auf die Lektüre von [65] verwiesen.

3.4.2. Log-basierte Verifikation

Mittels der Log-basierten Verifikation kann ein Ereignisprotokoll mit einem Prozessmodell in Form von definierten Eigenschaften über den Prozess verglichen werden. Es wird überprüft, ob die aufgezeichneten Instanzen des Ereignisprotokolls diesen definierten Eigenschaften entsprechen.

Das Verfahren von van der Aalst et al. [4] setzt Formeln auf der Grundlage linearer temporaler Logik (*linear temporal logic*, LTL) ein, um Eigenschaften über Instanzen (Fälle oder Traces) und Ereignisse des Prozeses festzulegen. Die von den Autoren entwickelte Sprache beinhaltet Elemente der Aussagenlogik, universelle und existenzielle Quantifizierung sowie temporale Operatoren wie *nexttime* (`_()`), *eventually* (`<>`), *always* (`[]`) und *until* (`F _U G`). Tabelle A.1 im Anhang A bietet eine Übersicht über die Bestandteile dieser Sprache. Auf Grundlage dieser Bestandteile lassen sich sich die Prozesseigenschaften in Bezug auf ein Ereignisprotokoll definieren, d. h. die Sprachelemente erlauben es, Eigenschaften über einzelne Instanzen und den zugehörigen Ereignissen zu formulieren. Während der Gegenüberstellung werden die Instanzen des Ereignisprotokolls der Reihe nach mit den in Formeln festgelegten Eigenschaften verglichen. Dabei werden jeweils die einzelnen Ereignisse (hier *Audit Trail Entry*, ATE genannt) der aktuellen Instanz der Reihe nach betrachtet.

Listing 3.1: Demonstration von LTL Formeln.

```
1  # Defining attributes
2  set pi.sachbearbeiter;
3  set ate.WorkflowModelElement;
4  set ate.Originator;
5  number ate.betrag;
6  date ate.Timestamp := "yyyy-MM-dd, HH:mm:ss";
7
8  # Renamings
9  rename ate.WorkflowModelElement as aktivitaet;
10 rename ate.Originator as teilnehmer;
11 rename ate.betrag as betrag;
12 rename ate.Timestamp as zeitstempel;
13 rename pi.sachbearbeiter as sachbearbeiter;
14
15 formula eigenschaft_1 () :=
16 {
17    <h2>Der Betrag eines Antrags ist nicht kleiner gleich Null.</h2>
18 }
19 !(<>(( aktivitaet == "Antrag registrieren" /\ betrag <= 0 )));
20
21 formula eigenschaft_2 () :=
22 {
23    <h2>Kein anderer Teilnehmer als der Sachbearbeiter eines Falls
24        schliesst einen Antrag ab. </h2>
25 }
26 !(<>(( aktivitaet == "Antrag abschliessen" /\
27                                  teilnehmer!=sachbearbeiter )));
28
29 formula eigenschaft_3 ( T: zeitstempel, U: zeitstempel ) :=
30 {
31    <h2>Alle Ereignisse besitzen einen Zeitstempel der zwischen
32        T und U liegt. </h2>
33 }
34 []( ( zeitstempel > T /\ zeitstempel < U ) );
```

Das Listing 3.1 demonstriert, wie Eigenschaften über das in Abschnitt 3.1 eingeführte Ereignisprotokoll (Tabelle 3.1) festgelegt werden können. Mit den Zeilen 2-6 werden die Typen der zu verwendenden Datenfelder definiert. Dabei kann mit `pi.sachbearbeiter` auf das Attribut $\#_{Sachbearbeiter}(i)$ der aktuellen Instanz i zugegriffen werden, `ate.betrag` liefert den Wert des Attributs $\#_{Betrag}(e_i)$ für das Ereignis e der Instanz i zurück. Um den Zugriff auf die Datenfelder bzw. Attribute zu erleichtern, findet in den Zeilen 9-13 eine Umbenennung statt. Ein erstes Beispiel für eine Formel wird in den Zeilen 15-19 gegeben. Die Formel mit dem Namen `eigenschaft_1` besagt, dass in der aktuellen Prozessinstanz kein Ereignis der Aktivität *Antrag registrieren* auftauchen darf, für welches das Attribut $\#_{Betrag}$ einen Wert kleiner oder gleich Null besitzt. Die Formel `eigen-`

`schaft_2` (Zeilen 21-27) vergleicht das aktuelle Instanzattribut $\#_{Sachbearbeiter}$ mit dem Teilnehmer eines Ereignisses. Es gilt, dass kein anderer Teilnehmer als der Sachbearbeiter eines Falls (bzw. einer Instanz) die Aktivität *Antrag abschließen* ausführen darf. Das letzte Beispiel für eine Formel in den Zeilen 29-34 demonstriert die Verwendung von Zeitstempeln (Typ `date`). Die Formel `eigenschaft_3` verlangt, dass alle Ereignisse im Ereignisprotokoll einen Zeitstempel besitzen, der zwischen dem Datum T und dem Datum U liegt, wobei zur Überprüfung der Formel Werte für T und U übergeben werden müssen.

In diesem Zusammenhang sei noch der Ansatz von Baumgrass et al. [21] erwähnt. Dieser Ansatz gibt einen Algorithmus an, mit welchem in einem XML-Format definierte Modelle für die rollenbasierte Zugriffskontrolle (RBAC) in LTL-Formeln transformiert werden können. Auf diese Weise kann der Aufwand für die Prüfung von durch RBAC definierte Prozesseigenschaften reduziert werden.

Das Verfahren zur Log-basierten Verifikation von Montali [58] verwendet zur Deklaration von Eigenschaften eine Sprache basierend auf Prädikatenlogik erster Stufe. Mit CLIMB (*Computational Logic for the verIfication and Modeling of Business constraints*) können Regeln zur Spezifikation von Prozessmodellen und Prozesseigenschaften angegeben werden, die anschließend mit dem Beweisverfahren SCIFF überprüft werden. Die Erstellung von Regeln erfolgt dabei auf Grundlage von Regelvorlagen, deren Syntax an natürliche Sprache angelehnt ist. Über die Elemente dieser Regelvorlagen, wie z. B. Aktivität, Teilnehmer und Zeitstempel, können Bedingungen eingefügt werden. Das folgende Beispiel demonstriert Bedingungen über Aktivitäten A und B sowie Teilnehmer O_A und O_B:

`IF activity` A `is` $performed$ `by` O_A
`having` A `equal to` $check$
`THEN activity` B `should NOT be` $performed$ `by` O_B
`having` B `equal to` $publish$ `and` O_B `equal to` O_A.

Mit dieser Regel lässt sich eine Sicherheitsanforderung zur Funktionstrennung (siehe auch Anforderung **(A7a)** in Abschnitt 2.3) überpüfen. Im Gegensatz zu dem zuvor vorgestellten LTL-basierten Verfahren [4] kann nicht auf Instanz- oder Ereignisattribute zugegriffen werden. Dafür ist es möglich, Bedingungen über Zeitstempel relativ zueinander und unter Angabe von Zeiteinheiten wie Minuten, Stunden oder Tagen zu formulieren.

So wie zuvor bei der Petri-Netz-basierten Konformitätsprüfung (Abschnitt 3.4.1) auf der Basis der Fitness, wird bei beiden Verfahren zur Log-basierten Verifikation auf Basis der in Formeln festgelegten Prozesseigenschaften die

Menge der Traces im Ereignisprotokoll in eine Menge konformer Traces L_{OK} (Eigenschaften sind erfüllt) und eine Menge nicht-konformer Traces L_{NOK} (eine oder mehrere Eigenschaften sind nicht erfüllt) aufgeteilt. Wenn $L_{NOK} = \varnothing$, dann sind die Prozesseigenschaften erfüllt. Anderenfalls enthält L_{NOK} Gegenbeispiele.

4. Entwurf der Fallstudien

Nach den Richtlinien von Runeson u. Höst [68] umfasst die Durchführung einer Fallstudie die folgenden fünf wesentlichen Phasen:

1. **Entwurf**: Definition der Ziele und Planung der Fallstudie
2. **Vorbereitung**: Definition der Prozedur für die Datensammlung
3. **Evidenzsammlung**: Ausführung der Datensammlung für das Fallbeispiel
4. **Analyse** der gesammelten Daten
5. **Bericht** über die Fallstudie

Demnach ist ein sorgfältiger Entwurf ein entscheidender Faktor für die Qualität von Fallstudien, so dass dieses Kapitel ganz der Entwurfsphase gewidmet werden soll. Zum Entwurf gehört neben der Definition des Ziels auch die Auswahl von Fallbeispielen (Abschnitt 4.1). Ebenso findet die Planung der Fallstudien statt. Diesbezüglich wird im Abschnitt 4.2 zur Methodik die Vorgehensweise vorgestellt. Dabei umfassen die Unterabschnitte sowohl die Vorbereitungsphase (4.2.1 und 4.2.2) als auch die Evidenzsammlungsphase (4.2.3). Der nach [68] geforderte Bericht wird in dieser Arbeit hauptsächlich durch Kapitel 4 als auch Kapitel 5 abgedeckt, wobei in Kapitel 5 die Beschreibung der Analysephase stattfindet.

4.1. Zieldefinition und Auswahl von Fallbeispielen

Das Ziel dieser Arbeit ist es, zu erkennen, ob sich die Überprüfung von Sicherheitsanforderungen (Abschnitt 2.3) mit Verfahren des Process Minings (Kapitel 3) praktisch umsetzen lässt. Dazu müssen für die Praxis repräsentative „Fälle der Studie“ bzw. Fallbeispiele ausgewählt werden, die sich auf einen konkreten Prozess beziehen. Zusätzlich müssen die erarbeiteten Sicherheitsanforderungen auf diesen Prozess angewendet werden können. Die Sicherheitsanforderungen **(A1a)** bis **(A11)** des Anforderungskatalogs können dabei in zwei Gruppen aufgeteilt werden, die sich auf jeweils einem Fallbeispiel betrachten lassen.

Die erste Gruppe beinhaltet die Sicherheitsanforderungen **(A1a)** bis **(A4)**, die sich auf Eigenschaften beziehen, die den Kontrollfluss betreffen (Verhalten

und funktionaler Aspekt). Für die Entwicklung eines Fallbeispiels für diese *Gruppe von strukturellen Sicherheitsanforderungen* spielt der Kontrollfluss bzw. die Struktur des Prozesses eine wichtige Rolle. Es sollen möglichst viele Ausführungsmuster beobachtet werden können. Dazu geeignet ist der **Prozess Schadenersatzanspruch**, der von Rozinat u. van der Aalst in [64] und [65] im Zusammenhang mit der Evaluation von Ergebnissen der Modell-Extraktion betrachtet wird. Dieser ist in Abb. 5.1 dargestellt. Es handelt sich um einen Prozess zur Abwicklung von Schadenersatzansprüchen in einer Versicherung (*liability insurance claim process*). Mit diesem wird eine fiktive (aber in der Praxis vorstellbare) Prozedur skizziert, die sowohl das wahlweise (XOR-Split/Join) als auch das nebenläufige (AND-Split/Join) Ausführungsmuster aufweist. Die beiden Ausführungsmuster stellen in der Praxis häufig auftretende und somit relevante Prozessstrukturen dar, die zudem zu den erarbeiteten Sicherheitsanforderungen passen.

Die Sicherheitsanforderungen der zweiten Gruppe hingegen beziehen sich nicht auf Eigenschaften über die Reihenfolge von Aktivitäten, sondern auf Eigenschaften einzelner oder mehrerer Ereignisse. Für ihre Überprüfung ist es daher unerheblich, welche Struktur der Prozess besitzt, in den sie eingebaut werden sollen. Die *nicht-strukturellen Sicherheitsanforderungen* **(A6a)** bis **(A11)** für den Organisations- und Datenaspekt bilden die zweite Gruppe. Die Grundlage für ein geeignetes Fallbeispiel bietet der **Prozess Kreditvergabe** von Arsac et al. [18]. Der Prozess zur Abwicklung von Kreditvergaben (*loan origination business process*) in einer Bank ist in Abb. 5.6 dargestellt. Die Struktur dieses Prozesses ist im Vergleich zum Prozess Schadenersatzanspruch einfach. Allerdings werden drei verschiedene Rollen eingeführt, über welche die Ausführung von Aktivitäten durch die Prozessteilnehmer festgelegt ist. Auch Zugriffe auf Datenobjekte werden durch den Prozess berücksichtigt, diese werden in diesem Fallbeispiel jedoch auf die vorliegenden Sicherheitsanforderungen angepasst.

4.2. Methodik

Abb. 4.1 veranschaulicht den methodischen Ablauf entlang der auf den Entwurf folgenden Phasen zur Vorbereitung, Evidenzsammlung und Analyse. Die folgenden Abschnitte gehen auf die methodischen Einzelheiten der Vorbereitung sowie der Evidenzammlung ein. Vorbereitende Schritte sind die Simulation (Abschnitt 4.2.1) sowie der Einbau von Abweichungen (Abschnitt 4.2.2). Die Anwendung des Process Minings zur Evidenzsammlung wird in Abschnitt 4.2.3 skizziert.

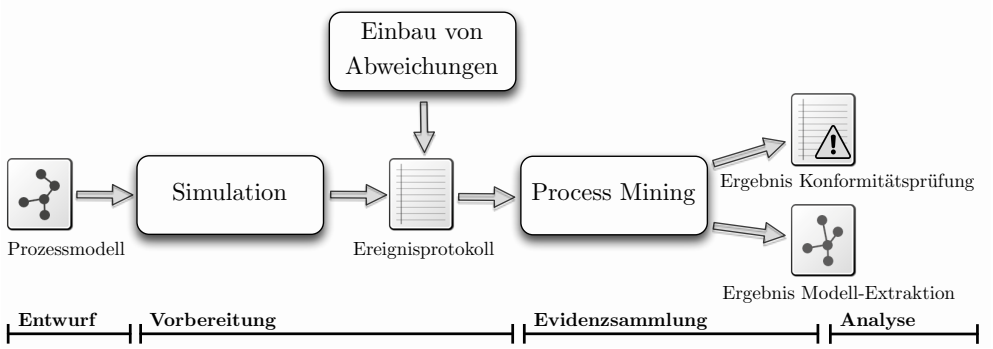

Abbildung 4.1.: Übersicht methodischer Ablauf entlang der Phasen Entwurf, Vorbereitung, Evidenzsammlung und Analyse.

4.2.1. Simulation

Um als Eingabe für das Process Mining Ereignisprotokolle zu erhalten, müssen ebensolche zuvor generiert werden, da für die speziell konstruierten Fallbeispiele kein Datenmaterial existiert. Dies kann durch eine Simulation erfolgen. Nach van der Aalst [2] muss die umfassende Simulation von Prozessen mindestens folgende Perspektiven und Eigenschaften modellieren:

a) **Kontrollfluss**, bestimmend die Reihenfolge von Aktivitäten mit Ausführungsmustern wie AND- und XOR-Split/Join sowie Schleifen

b) **Wahrscheinlichkeiten und Regeln**, bestimmen den Kontrollfluss bei Verzweigungen (z. B. Aktivität X bei Wert > 200, Aktivität Y sonst)

c) **Ressource/Organisation**, erweitert die Ausführung einer Aktivität um Ressourcen (z. B. Datensätze) und Teilnehmer (z. B. Benutzer, Rollen)

d) **Zeit**, Aktivitäten müssen den Zeitpunkt (diskret) oder den Intervall (kontinuierlich) ihrer Ausführung mit Zeitstempeln belegen

e) **Daten** (optional), zusätzliche Informationen zum betreffenden Fall (Instanz) oder der Aktivität

Es wurden verschiedene praktische Ansätze untersucht, um eine den Anforderungen entsprechende Lösung zu finden, z. B. CPN Tools [57, 67] und Process Log Generator (PLG) [26]. Das Werkzeug der Wahl für die Simulation wurde am IIG[1] entwickelt und ist Bestandteil des SWAT (*Security Workflow Analysis*

[1] Institut für Informatik und Gesellschaft, Albert-Ludwigs-Universität Freiburg

Toolkit) [14], einer Plattform zum Testen von Methoden zur Sicherheitsüberprüfung von Workflows. Das Simulationswerkzeug setzt sich aus mehreren Paketen von Java-Klassen zusammen, welche durch eine hierarchische Architektur und weitestgehende Kapselung einzelner Objekte und Funktionen die Möglichkeit bieten, die für die Simulation grundlegenden Parameter und Eigenschaften flexibel festzulegen. Hier soll die Funktionsweise der Simulation grob skizziert, sowie eine Auflistung der vorhandenen Parameter für die Konfiguration gegeben werden.

Als Grundlage für die Definition des Kontrollflusses verlangt das Simulationswerkzeug ein Prozessmodell in Form eines Petri-Netzes (reines S/T-Netz), das über das PNML-Format [48] eingebunden wird. Zusätzlich werden Wahrscheinlichkeiten für den Kantenübergang von einer Stelle zu mehreren Transitionen (XOR-Splits) angegeben. Weitere Eingaben betreffen die Eigenschaften der Perspektiven Ressource/Organisation, Daten und Zeit.

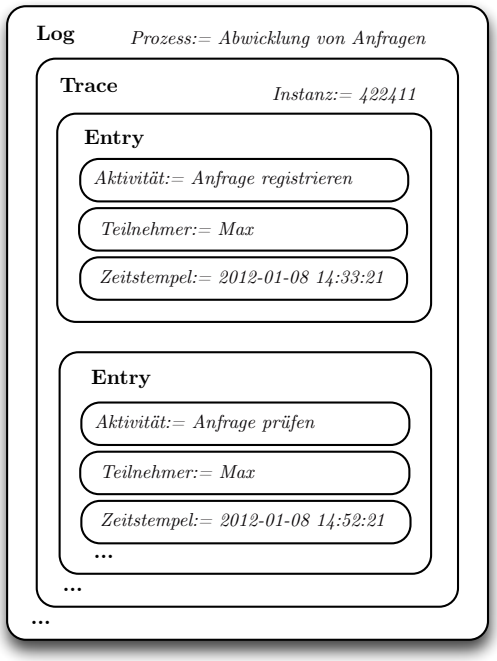

Abbildung 4.2.: Verschachtelter Aufbau eines Logs aus den Elementen Trace und Entry mit Attributen.

Das zu generierende Ereignisprotokoll (Log) besteht aus mehreren Elementen, die in Abb. 4.2 dargestellt werden. Ein Log zu einem Prozess enthält eine Reihe von Traces, die sich jeweils auf eine einzelne Prozessinstanz beziehen. Ein Trace ist aus Ereignissen, hier Entry (in Bezug auf den an anderer Stelle verwendeten Begriff *Audit Trail Entry*, ATE) aufgebaut. Ein Entry kennzeichnet jeweils die Ausführung einer Prozessaktivität durch einen Teilnehmer zu einem bestimmten Zeitpunkt. Weitere Attribute sind optionaler Bestandteil eines Entrys. Dieser Aufbau entspricht damit den Anforderungen an ein prozessorientiertes Ereignisprotokoll (siehe Abschnitt 3.1). Der in Pseudocode angegebene Algorithmus 1 skizziert den Ablauf der Simulation, um die dabei stattfindende Generierung eines Ereignisprotokolls zu veranschaulichen. Durch die wiederholte Positionierung einer Marke im Start-

Algorithmus 1: Ablauf der Simulation.

```
input  : Petri Netz P, Simulationsparameter S
output : Log L mit n Traces
initialize log L;
for i ⟵ 1 to n do
    initialize start state for P;
    initialize empty trace T ;
    while P has enabled transitions do
        fire an enabled transition in P;
        t ⟵ get next transition from petri net P;
        if t ≠ 0 and t is not silent then
            E ⟵ create entry for t based on S;
            use filters on entry E;
            add entry E to trace T;
use filters on trace T;
add trace T to log L;
write log L using log format;
```

zustand des eingebundenen Petri-Netzes und Anwendung der Feuerregel wird das Prozessmodell schrittweise simuliert. Für jede aktivierte Transition – und damit ausgeführte Aktivität – wird ein Entry mit der individuellen Konfiguration entsprechenden Attributen erzeugt. Versteckte Transitionen werden im Prozessmodell durch Verwendung eines mit Unterstrich beginnenden Bezeichners (z. B. `_H1`) markiert. Durch eine Abfrage wird sichergestellt, dass für versteckte Transitionen kein Entry erzeugt wird und diese somit nicht im Log auftauchen. Ist keine Transition mehr aktiv, wurde eine vollständige Sequenz von Aktivitäten durch das Prozessmodell, bzw. ein Trace, generiert. Durch den Einsatz von Filtern sowohl auf der Entry- als auch auf der Trace-Ebene ist es jeweils abschließend möglich, Entry- bzw. Trace-bezogene Eigenschaften an der passenden Stelle zu beeinflussen (*use filters on entry/trace*). Zuletzt wird das bisher nur intern repräsentierte Log mit seinen Elementen unter Verwendung des angegebenen Ausgabeformats in eine Datei geschrieben. Das konkrete Format des Ereignisprotokolls ist somit austauschbar.

Zur Simulation eines Ereignisprotokolls für ein Fallbeispiel wird die abstrakte Klasse `Simulation` erweitert und deren abstrakte Methoden implementiert, um die Simulationseigenschaften individuell festzulegen. Bei der Ausführung der Klasse wird basierend auf dieser Konfiguration ein dem Fallbeispiel entsprechen-

Listing 4.1: Ausschnitt aus einem Ereignisprotokoll im MXML-Format.

```
<?xml version="1.0" encoding="UTF-8"?>
<WorkflowLog>
  <Process id="Abwicklung von Anfragen">
    <ProcessInstance id="422411">
      <AuditTrailEntry>
        <WorkflowModelElement>Anfrage registrieren</
            WorkflowModelElement>
        <Timestamp>2012-01-08T14:33:21.713+0100</Timestamp>
        <Originator>Max</Originator>
      </AuditTrailEntry>
      <AuditTrailEntry>
        <WorkflowModelElement>Anfrage pruefen</WorkflowModelElement>
        <Timestamp>2012-01-08T14:52:21.131+0100</Timestamp>
        <Originator>Max</Originator>
        <Data>
          <Attribute name="...">...</Attribute>
        <Data>
      </AuditTrailEntry>
      ...
    </ProcessInstance>
    ...
  </Process>
</WorkflowLog>
```

des Ereignisprotokoll im MXML-Format erstellt[2]. Das Listing 4.1 zeigt einen entsprechenden Ausschnitt aus einem Ereignisprotokoll im MXML-Format zu den Beispieldaten in Abb. 4.2.

Eine Zusammenfassung der möglichen Eingaben und Konfigurationsparameter für die Simulation ist in Tabelle 4.1 dargestellt.

4.2.2. Einbau von Abweichungen

Die erarbeiteten Sicherheitsanforderungen werden in den Fallbeispielen entsprechend in Form von Bedingungen umgesetzt (siehe Kapitel 5). Der Verstoß gegen eine Bedingung bzw. das Nicht-Einhalten einer Sicherheitsanforderung wird im Folgenden *Abweichung*[3] genannt. Die an der Abweichung beteiligten Elemente des Prozesses werden als *abweichend* oder *nicht-konform* bezeichnet. Die Überprüfung auf Einhaltung von Sicherheitsanforderungen entspricht hier einer Suche

[2] Aus Platzgründen wird der Quelltext der Klassen sowie die generierten Ereignisprotokolle zu den Fallbeispielen nicht in der Arbeit abgedruckt. Alle relevanten Dateien befinden sich auf dem beigefügten Datenträger.

[3] In verwandten Arbeiten wird eine Abweichung auch Anomalie, (Sicherheits-)Verletzung oder Schwachstelle genannt.

Tabelle 4.1.: Übersicht über die Eingaben und Konfigurationsparameter für die Simulation eines Ereignisprotokolls mit dem Simulationswerkzeug.

Generelle Angaben
◦ Datum des Beginns des Logs ◦ Anzahl der zu generierenden Traces ◦ Anzahl von vervollständigten Traces pro Tag (> 0, $< Anzahl\ Traces$) sowie der Anteil der davon abweichenden Traces ($[0; 1]$)
Kontrollfluss
◦ Prozessmodell als Petri-Netz (im PNML-Format) ◦ Übergangswahrscheinlichkeiten für Kanten an XOR-Splits des Prozessmodells (falls vorhanden)
Zeit
◦ Dauer einer Aktivität in Sekunden, Minuten, Stunden, Tagen, Wochen oder Jahren sowie eine entsprechende Abweichung ($[0; 1]$)
Ressource/Organisation
◦ Liste von Benutzern ◦ Liste von Rollen ◦ Zuordnung von Rollen zu Benutzern ◦ Zuordnung von Rollen zu Aktivitäten
Daten
◦ Attributname und Wert (falls Wertebereich, dann mit Angabe von Wahrscheinlichkeiten zur Auswahl eines Werts) ◦ Zuordnung von Attributen zu Aktivitäten
Erweitert
◦ Liste von Entry-Filtern ◦ Liste von Trace-Filtern

nach Abweichungen im Ereignisprotokoll. Damit die Abwesenheit von falsch negativen oder falsch positiven Ergebnissen bei dieser Suche verifiziert werden kann, werden durch die Simulation in Bezug auf die Sicherheitsanforderungen konforme Ereignisprotokolle generiert. Der Einbau von Abweichungen erfolgt manuell, so dass alle Verletzungen vor der Suche bekannt sind. Somit kann das Ergebnis der Suche im Hinblick auf falsch negative Ergebnisse richtig bewertet werden.

An dieser Stelle soll die Systematik beschrieben werden, nach welcher nicht-konforme Traces in die Ereignisprotokolle der Fallbeispiele eingebaut werden.

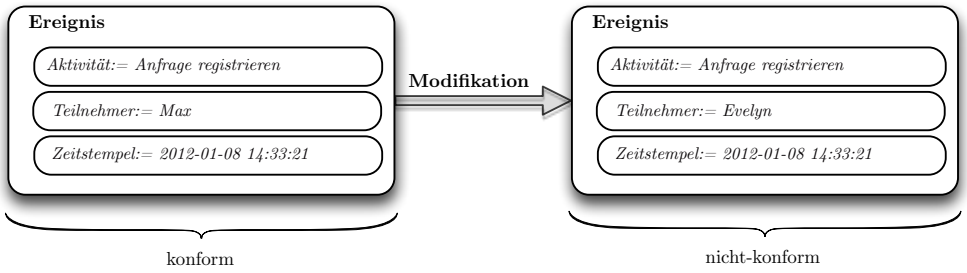

Abbildung 4.3.: Einbau einer Abweichung zur Bedingung „Aktivität *Anfrage registrieren* darf nur von Teilnehmer Max ausgeführt werden“.

Dies geschieht, indem durch die Simulation erzeugte und konforme Elemente des Ereignisprotokolls so angepasst werden, dass die der Sicherheitsanforderung zugrundeliegende Bedingung nicht mehr erfüllt ist. Für jede einzelne zu überprüfende Sicherheitsanforderung wird eine entsprechende Abweichung durch Modifikation bestehender Elemente eingebaut. Dies wird für einfache Sicherheitsanforderungen durch das Beispiel in Abb. 4.3 demonstriert.

Für die umfassendere Sicherheitsanforderung **(A3)** muss eine Reihe von Abweichungen vom vorgegebenen Kontrollfluss erzeugt werden. Dazu wird eine Aktivitätssequenz $\langle a_1, a_2, a_3, \ldots, a_{n-1}, a_n \rangle$ betrachtet, die den Ereignissen eines Trace σ im Ereignisprotokoll entspricht. Auf Grundlage einer solchen Aktivitätssequenz werden während des Einbaus durch die Anwendung der Operationen der Levenshtein-Distanz[4] folgende Formen von Abweichungen erzeugt:

- **Fehlende Aktivität** (*Löschen*): Hierbei wird eine Aktivität aus der Sequenz entfernt. Dabei ist darauf zu achten, an welcher Stelle die Aktivität entfernt wird: Die Sequenz $\langle a_1, a_2, a_3 \rangle$ kann durch Anwendung der *Löschen*-Operation zu einer Sequenz mit fehlendem Präfix ($\langle a_2, a_3 \rangle$), einer Sequenz mit fehlendem Suffix ($\langle a_1, a_2 \rangle$) oder zu einer Sequenz mit fehlendem Mittelteil werden ($\langle a_1, a_3 \rangle$).

[4]Die Levenshtein-Distanz beschreibt die Anzahl von nötigen Operationen, um das Alignment bzw. die Übereinstimmung zweier Sequenzen zu erreichen; die Operationen der Levenshtein- sowie der erweiterten Damerau-Levenshtein-Distanz werden hier nicht als Metrik sondern als Methode für die Umwandlung einer konformen Sequenz zu einer (nicht zwingend automatisch!) nicht-konformen Sequenz verwendet. Grundsätzlich lassen sich zwar alle möglichen Abweichungen ausschließlich unter Verwendung der Operationen *Löschen* (*deletion*) und *Einfügen* (*insertion*) herstellen, um jedoch typische Muster von Abweichungen zu erhalten, werden die Operationen *Ersetzen* (*substitution*) und *Vertauschen* (*swap*; Damerau-Levenshtein) ebenfalls angewendet.

- **Zusätzliche Aktivität** (*Einfügen*): Hierbei wird eine Aktivität zur Sequenz hinzugefügt. Es kann sich dabei einerseits um eine Aktivität handeln, die in dieser Sequenz bereits auftritt ($\langle a_1, a_2, a_3\rangle \rightarrow \langle a_1, a_2, a_2, a_3\rangle$; Dopplung) aber auch um eine andere Aktivität des Prozesses ($\langle a_1, a_2, a_3\rangle \rightarrow \langle a_1, a_2, a_x, a_3\rangle$). Die Stelle der *Einfüge*-Operation muss, wie bei der *Löschen*-Operation beschrieben, ebenfalls berücksichtigt werden.
- **Ersetzte Aktivität** (*Ersetzen*): Hierbei wird in einer Sequenz eine Aktivität a_i durch eine Aktivität a_j ersetzt, wobei $a_i \neq a_j$ gilt. Darüber hinaus gelten die gleichen Spezialfälle wie für die *Einfüge*-Operation beschrieben.
- **Vertauschte Aktivitäten** (*Vertauschen*): Hierbei werden zwei Aktivitäten innerhalb einer Sequenz miteinander vertauscht. Die betroffenen Aktivitäten a_i und a_j können nebeneinanderliegen ($\langle a_1, a_i, a_j, a_n\rangle$), oder auch weiter voneinander entfernt sein ($\langle a_i, a_2, a_j, a_n\rangle$).

Die hier vorgestellten Abweichungen vom vorgegebenen Kontrollfluss erschweren auch das Ziel von Verfahren der Modell-Extraktion, ein dem Prozess entsprechendes Prozessmodell zu finden (cf. Abschnitt 3.3.1 sowie Abschnitt 5.1.4). In diesem Kontext werden sie als Noise bezeichnet.

Während des Einbaus von Abweichungen im Ereignisprotokoll werden die betroffenen Traces anhand ihrer ID notiert. Eine vollständige Auflistung wird aus Gründen der Übersichtlichkeit ebenfalls bei der Beschreibung der Fallbeispiele in Kapitel 5 gegeben.

4.2.3. Evidenzsammlung mit ProM

Die Anwendung der in Kapitel 3 beschriebenen Verfahren zur Evidenzsammlung für die Fallbeispiele findet mit Hilfe des Rahmenwerks ProM[5] statt. Dieses bietet eine standardisierte Plattform für die Entwicklung und Anwendung von Process Mining-Techniken [35, 75]. ProM ist Open-Source-Software und wird von der Process Mining-Gruppe an der Technischen Universität Eindhoven (TU/e) verwaltet. Die ProM-Architektur basiert auf der Einbindung von Plugins, so dass Entwickler neuer Verfahren auf eine einheitliche Umgebung und gemeinsame Funktionen zurückgreifen können. ProM hat sich dadurch als de facto Standard für Process Mining etabliert, und seine Funktionalität, beziehungsweise die Menge von implementierten Techniken, wird von keinem anderen Process Mining-Produkt erreicht [3]. Die Anwendung erfordert jedoch Fachwissen, so dass sich die Zielgruppe aus fortgeschrittenen Endanwendern, Entwicklern und Systemadministratoren zusammensetzt.

[5] Homepage: `http://www.processmining.org/prom/start`

Frühere Versionen von ProM bis einschließlich ProM 5.2 (veröffentlicht 2009) verwenden MXML [34] als Eingabe-Format für Ereignisprotokolle. Ab Version 6 (veröffentlicht 2010) ist die primäre Unterstützung von Ereignisprotokollen im neuentwickelten und verbesserten XES-Format [43] vorgesehen, wobei das MXML-Format weiterhin eingelesen werden kann. Darüber hinaus können Plugins ab ProM 6 auch auf verteilten Rechnern ausgeführt werden. Die Benutzeroberfläche wurde grundlegend überarbeitet. Die genannten Änderungen machen es erforderlich, dass alle Plugins, die für frühere ProM Versionen geschrieben wurden, für ProM 6 neu implementiert werden müssen. Nur eine Teilmenge der 286 Plugins in ProM 5.2 wurde bisher reimplementiert. Die derzeit aktuelle Version 6.1 (veröffentlicht 2011) besitzt über 170 Plugins, darunter auch einige Neuentwicklungen. Für die Untersuchung im Rahmen dieser Arbeit werden sowohl ProM 5.2 als auch ProM 6.1 verwendet, um auch diejenigen Verfahren, die entweder nur in ProM 5.2 oder nur in ProM 6.1 umgesetzt wurden, in die Untersuchung einzuschließen. Durch die Simulation werden daher Ereignisprotokolle im MXML-Format erzeugt, so dass der Einsatz beider Versionen ohne vorangehende Konvertierung möglich ist.

Tabelle 4.2 gibt die in dieser Arbeit verwendeten Plugins an. Die Vorgehensweise bei der Anwendung der Plugins entspricht grob dem im Folgenden vorgestellten Ablauf:

1. **Laden des Ereignisprotokolls** in ProM. Je nach verwendetem Verfahren müssen gegebenenfalls weitere Eingaben in ProM geladen werden, z. B. ein Petri-Netz-Prozessmodell im PNML-Format für die Petri-Netz-basierte Konformitätsprüfung mit dem Plugin `Conformance Checker` oder eine Quelldatei mit LTL-Formeln für die Log-basierte Verifikation mit dem Plugin `LTL Checker`. Für die Verfahren zur Modell-Extraktion werden keine weiteren Eingaben benötigt.

2. **Aufruf des entsprechenden Plugins** für das anzuwendende Verfahren. Dabei werden hier stets die Standard-Parameter ausgewählt. Beim Plugin `SCIFF Checker` erfolgt die Eingabe der Bedingungen zu Sicherheitsanforderungen direkt im Plugin über die Konfiguration von Regelvorlagen. Für das Plugin `LTL Checker` ist dies ebenso für eine Reihe von Formelvorlagen möglich, es wird aber stets die Variante mit Angabe einer Quelldatei mit eigens erstellten Formeln gewählt.

3. **Ausführen des Plugins** zum Starten des Verfahrens mit anschließender Darstellung der Ergebnisse.

Tabelle 4.2.: Process Mining-Verfahren und eingesetzte ProM Plugins.

Verfahren	**ProM 5.2**	**ProM 6.1**
α-Algorithmus	`Alpha algorithm plugin`	-
Heuristisches Mining	`Heuristic Miner`	-
Genetisches Mining	`Genetic algorithm plugin`	-
Generierung eines expliziten Prozessmodells	`Explicit Model Miner`	-
Mining soziales Netzwerk	-	`Mine for a Working-Together Social Network`
Mining Rollenhierarchie	`Role Hierarchy Miner`	-
Petri-Netz-basierte Konformitätsprüfung	`Conformance Checker`	-
Log-basierte Verifikation nach van der Aalst et al. [4]	`LTL Checker`	`LTL Checker`
Log-basierte Verifikation nach Montali [58]	`SCIFF Checker`	-
Konvertierung RBAC-Modell zu LTL nach Baumgrass et al. [21]	-	`RBAC to LTL conversion`

Die Screenshots in den folgenden Abbildungen 4.4 bis 4.6 sollen einen Eindruck der Benutzeroberfläche der beiden Programmversionen sowie einiger Plugins vermitteln.

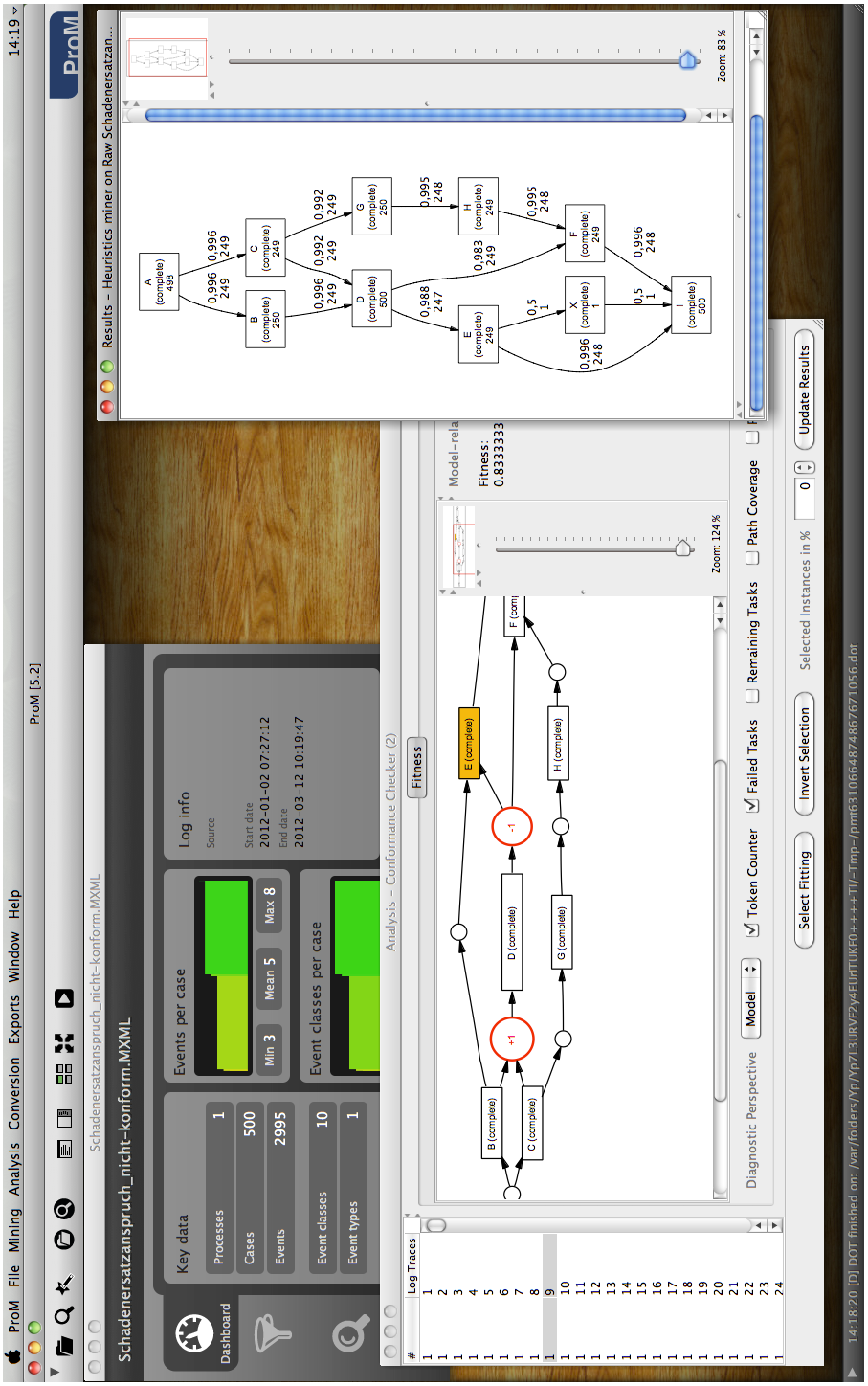

Abbildung 4.4.: ProM 5.2: Darstellung eines Ereignisprotokolls (oben), Ergebnisse der Plugins `Conformance Checker` (unten) und Heuristics Miner (rechts).

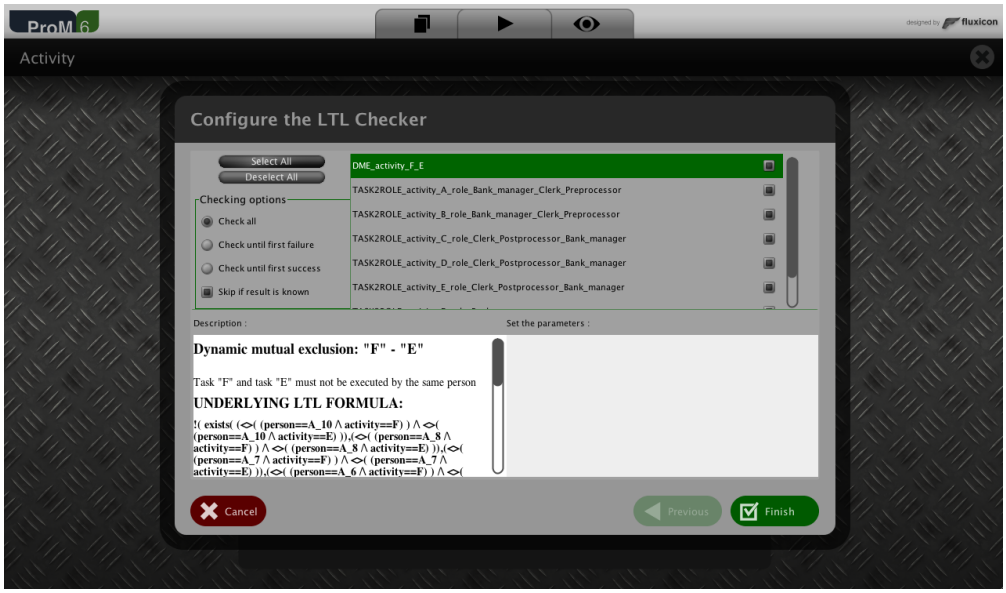

Abbildung 4.5.: ProM 6.1: Aufruf des Plugins LTL-Checker mit den LTL-Formeln zur Überprüfung von RBAC-Vorgaben.

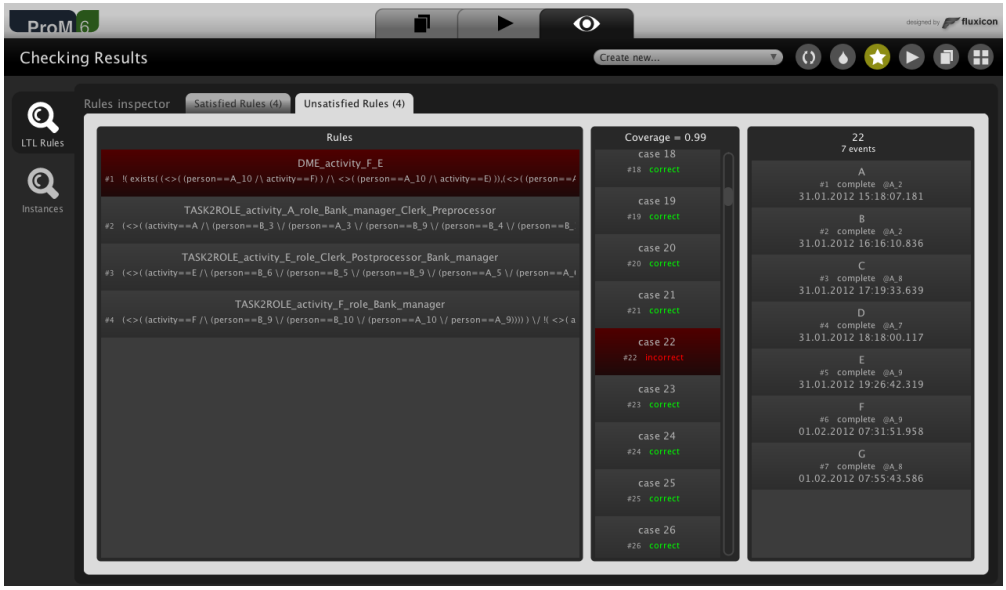

Abbildung 4.6.: ProM 6.1: Ergebnisse des Plugins LTL-Checker zu den LTL-Formeln zur Überprüfung von RBAC-Vorgaben.

5. Durchführung der Fallstudien

In diesem Kapitel wird die Durchführung der Fallstudien präsentiert. Dies umfasst sowohl die Evidenzsammlung als auch die Analyse der Fallbeispiele. Aus Gründen der Übersichtlichkeit erfolgt auch die Beschreibung der Fallbeispiele und der individuellen Vorbereitung in diesem Kontext. In den Abschnitten 5.1 und 5.2 werden die ausgewählten Fallbeispiele nacheinander betrachtet. Die entsprechenden Abschnitte sind jeweils in die Unterabschnitte Beschreibung, Vorbereitung und Abschnitten zur Überprüfung von Sicherheitsanforderungen aufgeteilt.

5.1. Prozess Schadenersatzanspruch

5.1.1. Beschreibung des Fallbeispiels

In diesem Fallbeispiel wird der von Rozinat [64] beschriebene Prozess zur Abwicklung von Schadenersatzansprüchen in einer Versicherung betrachtet. Abbildung 5.1 zeigt das Prozessmodell zu diesem Prozess als Petri-Netz. Das Ziel dieses

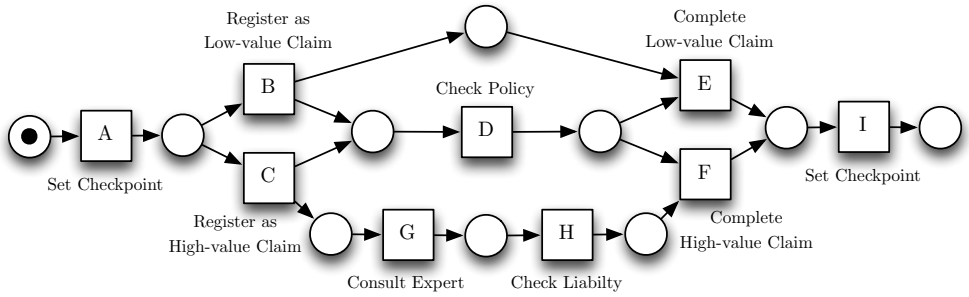

Abbildung 5.1.: Prozess zur Abwicklung von Schadenersatzansprüchen in einer Haftpflichtversicherung nach [64].

Prozesses ist es, das Betrugsrisiko bei Forderungen hoher Beträge zu mindern. Der Prozess setzt dieses Ziel um, indem Forderungen mit einem hohen Betrag (*high-value claims*) unter Einbezug eines Experten zusätzlich geprüft werden (Aktivitäten *Consult Expert* (G) und *Check Liability* (H)).

Tabelle 5.1.: Sicherheitsanforderungen zum Fallbeispiel Schadenersatzanspruch.

Name	Beschreibung
(A1a)	Aktivität *Reject Claim* (X) darf nicht enthalten sein.
(A1b)	Aktivität *Check Policy* (D) muss enthalten sein.
(A2a)	Wird Aktivität *Complete High-value Claim* (F) ausgeführt, muss zuvor Aktivität *Check Liability* (H) ausgeführt worden sein.
(A2b*)	Wird Aktivität *Register as High-value Claim* (C) ausgeführt, muss danach Aktivität *Consult Expert* (G) innerhalb von drei Tagen ausgeführt werden.
(A3)	Alle Prozessabläufe entsprechen den Aktivitätssequenzen, die durch das Prozessmodell in Abb. 5.1 vorgegeben werden: - $\langle A, B, D, E, I\rangle$ - $\langle A, C, D, G, H, F, I\rangle$ - $\langle A, C, G, D, H, F, I\rangle$ - $\langle A, C, G, H, D, F, I\rangle$
(A4)	Auf Aktivität *Set Checkpoint* (A) folgt Aktivität *Register as Low-value Claim* (B) falls das Instanzattribut $\#_{Betrag} < 5000$, Aktivität *Register as High-value Claim* (C) anderenfalls.

Tabelle 5.1 zeigt, wie die Gruppe der strukturellen Sicherheitsanforderungen (siehe Anforderungskatalog in Tabelle 2.4 und Abschnitt 2.3) in Form konkreter Bedingungen über die Prozessausführung in das Fallbeispiel integriert wurden.

5.1.2. Vorbereitung

Da die Simulation des Prozesses auf Grundlage des Prozessmodells in Abb. 5.1 vorgenommen wird, ist das resultierende Ereignisprotokoll in Hinblick auf die Sicherheitsanforderungen **(A1a)**, **(A1b)**, **(A2a)** sowie **(A3)**, welche enthaltene Aktivitäten und ihre Reihenfolgen betreffen, korrekt. Die Einhaltung der Sicherheitsanforderung **(A2b*)** wird dadurch sichergestellt, dass die Dauer der einzelnen Aktivitäten durch entsprechende Simulationsparameter auf 10 Minuten festgelegt ist. Die verwendeten Simulationsparameter für dieses Fallbeispiel sind im Anhang B.2 angegeben. Im Ereignisprotokoll muss für jede Prozessinstanz der Betrag der laufenden Forderung vermerkt sein, damit geprüft werden kann, ob während der Prozessausführung die entsprechende Verzweigung auf Basis des Betrags eingehalten wird (siehe Anforderung **(A4)**). Da das Simulationswerkzeug weder (a) Instanzattribute noch (b) Verzweigung auf Grundlage von Attributen (datenbasiertes Branching) unterstützt, wird das Ereignisprotokoll zunächst ohne ein entsprechendes Attribut für den Betrag erzeugt. Anschließend wird ein Perl-Skript (siehe Anhang B.1) angewendet,

Tabelle 5.2.: Abweichungen im Ereignisprotokoll zum Fallbeispiel Schadenersatzanspruch.

Trace	Abweichende Eigenschaften	Verletzte Sicherheitsanforderung(en)
5	$\langle A, B, D, E, X, I \rangle$	**(A1a)**, da *Reject Claim* (X) enthalten, **(A3)** außerdem
9	$\langle A, B, E, I \rangle$	**(A1b)**, da *Check Policy* (D) fehlt, **(A3)** außerdem
18	$\langle A, C, G, D, F, I \rangle$	**(A2a)**, da Aktivität H vor F fehlt, **(A3)** außerdem
25	$\#_{Zeitstempel}(C) = 2012-01-04$, $\#_{Zeitstempel}(G) = 2012-02-04$	**(A2b*)**, da zwischen Aktivitäten C und G mehr als 3 Tage liegen
29	$\langle B, D, E, I \rangle$	**(A3)**, Gelöschte Aktivität A
30	$\langle D, E, I \rangle$	**(A3)**, Gelöschte Aktivitäten A und B
31	$\langle A, B, D, I \rangle$	**(A3)**, Gelöschte Aktivität E
34	$\langle A, C, D, G, H, F \rangle$	**(A3)**, Gelöschte Aktivität I
43	$\langle A, C, D, D, G, H, F \rangle$	**(A3)**, Eingefügte Aktivität D
47	$\langle G, A, B, D, E, I \rangle$	**(A3)**, Eingefügte Aktivität G
52	$\langle A, B, D, E, I, I \rangle$	**(A3)**, Eingefügte Aktivität I
57	$\langle A, B, D, H, I \rangle$	**(A3)**, Ersetzte Aktivität E durch H
63	$\langle A, B, E, D, I \rangle$	**(A3)**, Vertauschen von Aktivitäten D und E
70	$\langle A, B, D, E, I \rangle$, $\#_{Betrag} = 200.000$	**(A4)**, da Betrag nicht einem *Low-value Claim* (B) entspricht
78	$\langle A, C, D, G, H, F, I \rangle$, $\#_{Betrag} = 350$	**(A4)**, da Betrag nicht einem *High-value Claim* (C) entspricht

um das Instanzattribut $\#_{Betrag}$ nachträglich einzusetzen. Dabei entspricht der zufällig erzeugte Wert für den Betrag dem Verarbeitungspfad der betrachteten Instanz der in **(A4)** angegebenen Schranke, d. h. konkret: wird in einer Instanz die Aktivität *Register as Low-value claim* (B) ausgeführt, so wird ein Wert von 1-4999 gewählt. Wird die Aktivität *Register as High-value claim* (C) ausgeführt, so wird ein Wert von 5000-500000 gewählt. Dadurch wird sichergestellt, dass das durch das Ereignisprotokoll dokumentierte Verhalten den Anforderungen an die Prozessausführung entspricht, auch wenn eine entsprechende Simulation nicht durchgeführt werden konnte.

In das bisher zu den Sicherheitsanforderungen des Fallbeispiels konforme Ereignisprotokoll werden anschließend mit der in Abschnitt 4.2.2 beschriebenen Systematik Abweichungen eingebaut. Tabelle 5.2 listet die abweichenden Instanzen des Ereignisprotokolls zusammen mit ihren Eigenschaften und den durch sie verletzten Sicherheitsanforderungen auf. Die Sicherheitsanforderung **(A3)** schließt dabei implizit auch die Sicherheitsanforderungen **(A1a)**, **(A1b)** sowie

(A2a) ein, so dass Abweichungen von diesen drei Sicherheitsanforderungen gleichzeitig auch die Sicherheitsanforderung **(A3)** verletzen.

Das gemäß dieser Beschreibung vorbereitete Ereignisprotokoll dient in diesem Fallbeispiel auf zweierlei Art als Grundlage für die Überprüfung der Sicherheitsanforderungen. Eine *direkte Überprüfung* findet durch die Verfahren zur Konformitätsprüfung statt, die direkt auf einem Ereignisprotokoll angewendet werden (Abschnitt 5.1.3). Ob auch eine *indirekte Überprüfung* auf Grundlage der durch Verfahren zur Modell-Extraktion aus dem Ereignisprotokoll gewonnenen *de facto* Prozessmodellen möglich ist, wird in anschließend in Abschnitt 5.1.4 evaluiert.

5.1.3. Direkte Überprüfung der Sicherheitsanforderungen

Die in Kapitel 3 vorgestellten Verfahren zur Konformitätsprüfung sind in ProM 5.2 durch die Plugins `LTL Checker` bzw. `SCIFF Checker` (Log-basierte Verifikation nach van der Aalst et al. [4] bzw. Montali [58]) sowie `Conformance Checker` (Petri-Netz-basierte Konformitätsprüfung nach Rozinat u. van der Aalst [65]) umgesetzt. Ihre Anwendung zur direkten Überprüfung der Sicherheitsanforderungen auf dem Ereignisprotokoll, bzw. zur Suche nach den enthaltenen Abweichungen wird in den folgenden Absätzen beschrieben.

LTL Checker. Zur Überprüfung der Sicherheitsanforderungen **(A1a)**, **(A1b)**, **(A2a)**, **(A3)** und **(A4)** mit der Log-basierten Verifikation nach van der Aalst et al. [4] werden entsprechende LTL-Formeln (siehe Listing C.1, aus Platzgründen in Anhang C) vorbereitet. Für die Sicherheitsanforderung **(A2b*)** konnte keine entsprechende LTL-Formel erstellt werden, da im Sprachumfang (siehe Anhang A) keine arithmetischen Operatoren enthalten sind, die im Zusammenhang mit dem erforderlichen Vergleich der Zeitstempel benötigt werden. Das Ergebnis des Plugins `LTL Checker` teilt die Menge T_{ALL} mit den 500 Traces des Ereignisprotokolls in eine Menge konformer Traces T_{OK} und eine Menge nicht-konformer Traces T_{NOK} auf. T_{NOK} enthält jeweils genau die entsprechenden abweichenden Traces aus Tabelle 5.2, beispielsweise liefert die Anwendung der LTL-Formel zur Sicherheitsanforderung **(A3)** die Menge $T_{NOK} = \{5, 9, 18, 29, 30, 31, 34, 43, 47, 52, 57, 63\}$. Mit Ausnahme der abweichenden Traces zur Sicherheitsanforderung **(A2b*)**, die sich aufgrund des beschränkten Sprachumfangs nicht als Formel ausdrücken lässt, können mit dem `LTL Checker` alle restlichen abweichenden Traces aus Tabelle 5.2 identifiziert werden.

SCIFF Checker. Für die Überprüfung mit dem `SCIFF Checker` müssen die Sicherheitsanforderungen anhand der bestehenden Regelvorlagen konfiguriert

Tabelle 5.3.: Konfiguration der Regelvorlagen des Plugins `SCIFF Checker` zur Überprüfung der Sicherheitsanforderungen des Fallbeispiels Schadenersatzanspruch.

Anforderung	Regel
(A1a)	`IF activity` A `is` *performed* `by` O_A `at time` T_A `having` A `equal to` X `THEN false`
(A1b)	`activity` A `should be` *performed* `by` O_A `at time` T_A `having` A `equal to` D
(A2a)	`IF activity` A `is` *performed* `by` O_A `at time` T_A `having` A `equal to` F `THEN activity` B `should be` *performed* `by` O_B `at time` T_A `having` B `equal to` H `and` T_B `before` T_A
(A2b*)	`IF activity` A `is` *performed* `by` O_A `at time` T_A `having` A `equal to` C `THEN activity` B `should be` *performed* `by` O_B `at time` T_A `having` B `equal to` G `and` T_B `before` $T_A + [3d, 0ms]$

werden. Dies gelingt nur für die Sicherheitsanforderungen **(A1a)**, **(A1b)**, **(A2a)** sowie **(A2b*)** durch die in Tabelle 5.3 beschriebenen Regeln. Die Sicherheitsanforderung **(A3)** kann auf Grundlage der bestehenden Regelvorlagen nicht entsprechend umgesetzt werden. Für die Formulierung der Sicherheitsanforderung **(A4)** ist der Zugriff auf Instanzattribute notwendig, was jedoch durch das Plugin nicht vorgesehen ist. Bei der Anwendung der Regeln auf dem Ereignisprotokoll liefert der SCIFF Checker genau wie der LTL Checker eine Menge konformer Traces T_{OK} sowie eine Menge nicht-konformer Traces T_{NOK}. Mit Ausnahme der abweichenden Traces zu den Sicherheitsanforderungen **(A3)** und **(A4)** für die keine entsprechenden Regeln aus den vorhandenen Regelvorlagen gebildet werden können, identifiziert der `SCIFF Checker` alle restlichen abweichenden Traces aus Tabelle 5.2 korrekt.

Conformance Checker. Die Petri-Netz-basierte Konformitätsprüfung wird angewendet, um die Sicherheitsanforderung **(A3)** und implizit auch **(A1a)**, **(A1b)** sowie **(A2a)** zu überprüfen. Die Sicherheitsanforderungen **(A2b*)** sowie **(A4)** können nicht überprüft werden, da für diese über die reine Abfolge von Aktivitäten hinaus auch Zeitstempel und Instanzattribute berücksichtigt werden

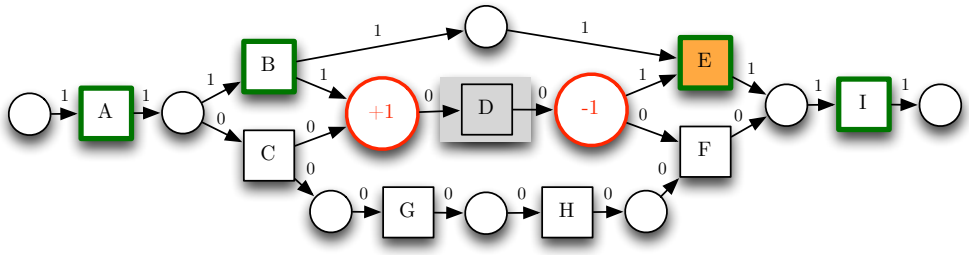

Abbildung 5.2.: Visualisierung des Replays von Trace 9 mit der Aktivitätssequenz $\langle A, B, E, I \rangle$ auf dem Prozessmodell des Fallbeispiels.

müssen, was das Verfahren nicht leistet. Zunächst wird das in Abb. 5.1 dargestellte Prozessmodell im PNML-Format in ProM geladen und die Transitionen des Petri-Netzes mit den entsprechenden Ereignissen im Ereignisprotokoll verknüpft. Dabei werden nur Ereignisse berücksichtigt, deren Aktivität als Transition im Petri-Netz enthalten ist, so dass das Ereignis zur Aktivität *Reject claim* (X) in der Instanz 5 ignoriert wird und folglich diese Abweichung unerkannt bleiben muss. Nach der Verknüpfung von Ereignisprotokoll und Prozessmodell wird das Plugin `Conformance Checker` eingesetzt. Auf Grundlage des Replays aller Traces im Ereignisprotokoll auf dem gegebenen Prozessmodell leistet das Plugin Folgendes:

- Berechnung der Marken-basierten Fitness f_{token} für einzelne Traces (siehe Tabelle 5.4) sowie für das gesamte Ereignisprotokoll (0.966195).
- Visualisierung des Replays auf dem Prozessmodell (siehe Abb. 5.2). Dadurch wird eine Identifikation der Art der Abweichung ermöglicht.
- Aufteilung der Menge aller Traces T_{ALL} in eine Menge nicht-konformer Traces T_{NOK} mit $f_{token} < 1$ und eine Menge konformer Traces $T_{OK} = T_{ALL} \setminus T_{NOK}$ mit $f_{token} = 1$.

Für die Überprüfung der Sicherheitsanforderung **(A3)** im Sinne einer reinen Identifikation von Abweichungen ist die Marken-basierte Fitness f_{token} nur insofern von Bedeutung, als dass die Klassifikation der Traces auf ihrer Basis vorgenommen wird. Allerdings kann die Fitness f_{token} darüber hinaus in einem gewissen Rahmen auch den Schweregrad der Abweichung reflektieren, d. h. je geringer f_{token}, desto größer ist der strukturelle Unterschied zwischen einem Trace mit Fitness f und dem vorgegebenen Prozessmodell. So gilt für den abweichenden Trace 29 (Aktivitität A gelöscht) und Trace 30 (Aktivität A und B gelöscht): $f(_{token}29) = 0.833333 > f_{token}(30) = 0.675$. Demnach drückt die

Tabelle 5.4.: Fitness-Werte der nicht-konformen Traces als Ergebnis der Petri-Netz-basierten Konformitätsprüfung.

Trace	Fitness f_{token}
9	0.833333
18	0.875
29	0.833333
30	0.675
31	0.733335
34	0.875
43	0.9
47	0.875
52	0.875
57	0.66964287
63	0.85714287

vergleichende Bewertung mit der Fitness f_{token} deutlich aus, dass das Fehlen einer einzelnen Aktivität weniger schwerwiegende Folgen für das Replay hat als das Löschen zweier Aktivitäten.

Wie bereits erwähnt, kann durch die Visualisierung eines Trace auch die Art dessen Abweichung genauer bestimmt werden. Abb. 5.2 zeigt die Visualisierung des Replays von Trace 9 mit der Aktivitätssequenz $\langle A, B, E, I\rangle$ auf dem Prozessmodell[1]. Die Beschriftung der Kanten zeigt an, welche Kanten während des Replays benutzt wurden (1: benutzt, 0: unbenutzt), die Transitionen A, B, E, und I (grün umrandet) wurden entsprechend der Aktivitätssequenz gefeuert. Da in der Sequenz die Aktivität D nicht auftaucht, wird auch die Transition D im Prozessmodell nicht gefeuert (grau hinterlegt), und die eingehende Marke (rot; +1) wird von D nicht konsumiert. Aus diesem Grund produziert D auch keine ausgehende Marke (rot; -1), und für die Fortsetzung des Replays muss die in der Farbe Orange hinterlegte Transition E künstlich gefeuert werden. Zusammengefasst lässt sich daraus schließen: in Trace 9 fehlt die Aktivität D.

Die Betrachtung der Menge nicht-konformer Traces T_{NOK} = {9, 18, 29, 30, 31, 34, 43, 47, 52, 57, 63} zeigt, dass mit Ausnahme von Trace 5 (verletzt **(A1a)** und **(A3)**), der wie oben beschrieben nicht vollständig eingelesen werden konnte, die weiteren Abweichungen der Sicherheitsanforderungen **(A1b)**, **(A2a)** sowie **(A3)** mit dem `Conformance Checker` identifiziert werden können.

[1] Aus Platzgründen wurde auf eine Einbindung der sehr in die Breite gehenden Originaldarstellung verzichtet. Die hier gezeigte Darstellung entspricht jedoch in Farbe und Form exakt der Visualisierung die von ProM 5.2 mit dem Plugin `Conformance Checker` erzeugt wird.

Tabelle 5.5.: Verfahren der Konformitätsprüfung und Sicherheitsanforderungen: Anteil der identifizierten Abweichungen des Fallbeispiels Schadenersatzanspruch.

	(A1a)	**(A1b)**	**(A2a)**	**(A2b*)**	**(A3)**	**(A4)**
Log-basierte Verifikation (`LTL Checker`)	1/1	1/1	1/1	-	12/12	2/2
Log-basierte Verifikation (`SCIFF Checker`)	1/1	1/1	1/1	1/1	-	-
Petri-Netz basierte Konformitätsprüfung (`Conformance Checker`)	-	1/1	1/1	-	11/12	-

Zusammenfassung. Abschließend bietet Tabelle 5.5 einen Überblick über das Ergebnis der Überprüfung der Sicherheitsanforderungen mit den in diesem Abschnitt betrachteten Verfahren. Zusammengefasst ergeben sich dabei folgende Erkenntnisse:

- Zusammengenommen sind die Verfahren der Log-basierten Verifikation für die Überprüfung aller betrachteten Sicherheitsanforderungen geeignet. Die individuellen Unzulänglichkeiten, die in diesem Abschnitt offengelegt werden, sind rein technischer Natur und können somit durch eine verbesserte Implementierung der Verfahren behoben werden.
- Mit der Petri-Netz-basierten Konformitätsprüfung lassen sich Sicherheitsanforderungen **(A2b*)** und **(A4)** grundsätzlich nicht überprüfen, da die nötigen Informationen nicht durch Petri-Netz-Prozessmodelle repräsentiert werden. Allerdings sollte es kein Problem darstellen, das Verfahren so zu erweitern, dass Aktivitäten aus dem Ereignisprotokoll, die nicht im Modell auftreten, nicht ignoriert werden. Dadurch kann die Überprüfung von Sicherheitsanforderungen wie **(A1a)** ermöglicht werden.

5.1.4. Indirekte Überprüfung der Sicherheitsanforderungen

Dieser Abschnitt beschreibt die indirekte Überprüfung von Sicherheitsanforderungen auf Grundlage von durch Verfahren der Modell-Extraktion gewonnenen *de facto* Prozessmodellen, die den Kontrollfluss beschreiben. Wie es sich bereits aus der Beschreibung dieser Verfahren in Kapitel 3 folgern lässt, bieten die durch sie konstruierten Modelle keine zuverlässige Basis für die Überprüfung von Sicherheitsanforderungen. Dies soll für die in Abschnitt 3.3.1 beschriebenen Verfahren des α-Algorithmus sowie dem heuristischen und genetischen Mining im Folgenden praktisch demonstriert werden.

Zur Erstellung der aus der Anwendung der genannten Verfahren resultierenden *de facto* Prozessmodelle wird das Ereignisprotokoll in ProM 5.2 geladen und die entsprechenden Plugins `Alpha algorithm plugin`, `Heuristic Miner` sowie `Genetic algorithm plugin` unter Verwendung der Standard-Parameter ausgeführt. Anschließend werden die sich aus dem Heuristic Mining sowie dem Genetic Mining ergebenden heuristischen Netze direkt in ProM in Petri-Netze konvertiert, um bessere Vergleichsmöglichkeiten zu erzielen. Abbildung 5.3 zeigt die Prozessmodelle, die von diesen Plugins auf Grundlage des Ereignisprotokolls erzeugt werden. Wie im vorangegangenen Abschnitt bei der Petri-Netz-basierten

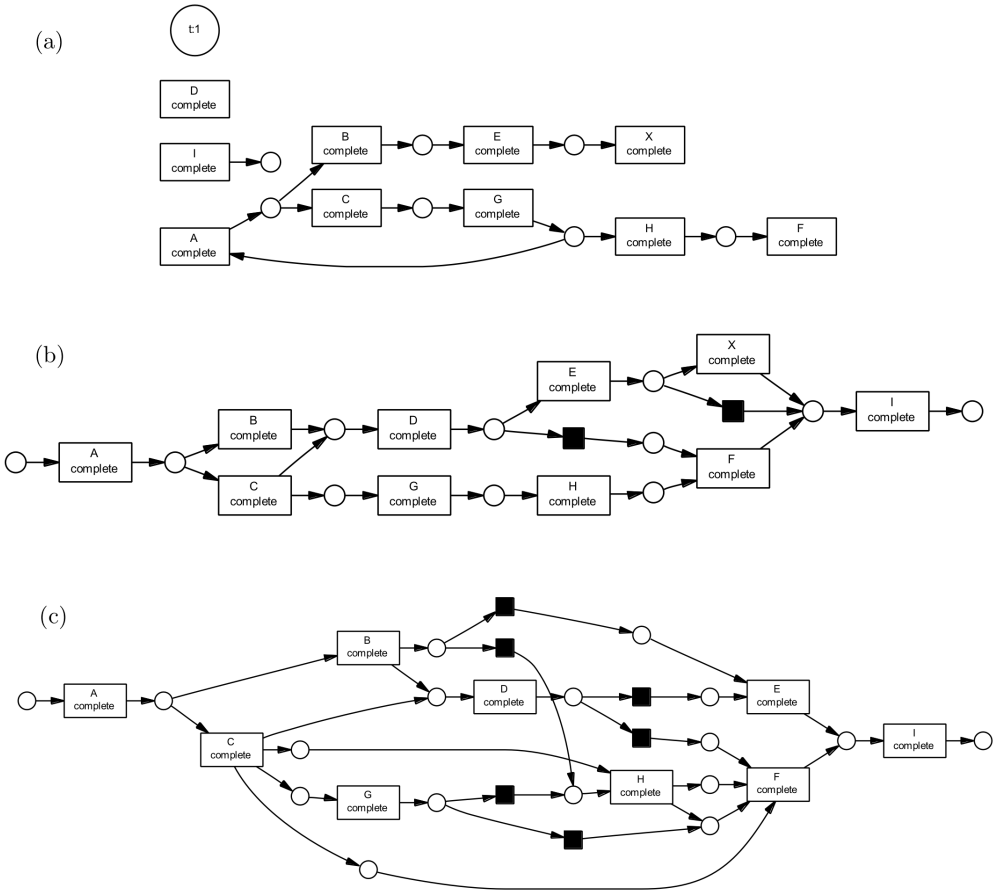

Abbildung 5.3.: Mit ProM 5.2 extrahierte *de facto* Prozessmodelle: (a) `Alpha algorithm plugin`, (b) `Heuristic miner` und (c) `Genetic algorithm plugin`.

Konformitätsprüfung sind die Sicherheitsanforderungen **(A2b*)** sowie **(A4)** durch die Betrachtung dieser Prozessmodelle nicht greifbar, da diese keine Informationen über Zeitstempel oder Instanzattribute enthalten. Es wird daher nur untersucht, ob sich die Abweichungen der Sicherheitsanforderungen **(A1a)**, **(A1b)**, **(A2a)** und **(A3)** in diesen Prozessmodellen erkennen lassen. Dabei ist vorrangig von Interesse, ob falsch negative Ergebnisse entstehen (Abweichungen werden nicht durch das Modell repräsentiert), aber auch, ob falsch positive Ergebnisse zu erwarten sind (das Modell weist Pfade entsprechend Abweichungen auf, die nicht im Ereignisprotokoll vorhanden sind).

(A1a) und (A1b). Die Abweichung von Sicherheitsanforderung **(A1a)** drückt sich durch das Enthaltensein der Transition X in den Prozessmodellen (a) und (b) klar aus. Lediglich das Prozessmodell (c) des Genetic Mining bildet die Prozessaktivität X aus dem Ereignisprotokoll nicht ab, so dass die Abweichung auf diesem Prozessmodell unerkannt bleibt. Es wird an dieser Stelle auch unmittelbar deutlich, dass durch die Reduktion des Ereignisprotokolls auf ein Prozessmodell die Information verlorengeht, welcher Trace oder welche Traces für die Abweichung verantwortlich sind. Dies kann bei der hier vorgestellten indirekten Überprüfung der Sicherheitsanforderung im Gegensatz zu der im letzten Abschnitt beschriebenen direkten Überprüfung mit Verfahren der Konformitätsprüfung prinzipiell nicht erkannt werden. Die Sicherheitsanforderung **(A1b)** drückt aus, dass in jedem möglichen Pfad durch das Prozessmodell die Aktivität bzw. Transition D enthalten sein muss. Dies ist für die Prozessmodelle (b) und (c) der Fall, woraus zu schließen wäre, dass die Traces des Ereignisprotokolls im Hinblick auf **(A1b)** konform sind. Dies ist jedoch nicht korrekt, da das Ereignisprotokoll den von **(A1b)** abweichenden Trace 9 enthält (siehe Tabelle 5.2). Somit wird an dieser Stelle erneut aufgezeigt, dass Abweichungen auf den extrahierten Prozessmodellen unerkannt bleiben können.

(A2a). Zur Überprüfung der Sicherheitsanforderung **(A2a)** (tritt Aktivität F auf, muss Aktivität H vorher ausgeführt worden sein) wird die Transition F in den Prozessmodellen betrachtet. In allen drei Fällen besitzt diese Transition eine eingehende Stelle, für die ausschließlich von der Transition H eine Marke produziert werden kann. Somit gilt, dass die Transition F nur dann aktiviert werden kann, wenn H zuvor aktiviert worden ist. Somit reflektiert keines der drei Prozessmodelle die im Ereignisprotokoll enthaltene Abweichung zu dieser Sicherheitsanforderung.

(A3). Im Zusammenhang mit der Überprüfung dieser Sicherheitsanforderung sind alle Abweichungen von den vorgegebenen Aktivitätsreihenfolgen von Interesse. Um zu erkennen, ob die drei Prozessmodelle im Hinblick auf das Ereignisprotokoll vollständig sind, d. h. ob alle Traces des Ereignisprotokolls einschließlich der abweichenden Traces durch das Prozessmodell reflektiert werden, kann statt einer manuellen Prüfung der Prozessmodelle, wie in den letzten Absätzen beschrieben, auch die Petri-Netz-basierte Konformitätsprüfung mit dem Plugin `Conformance Checker` durchgeführt werden. Dabei wird nicht die durch das Plugin berechnete Marken-basierte Fitness f_{token} (siehe Definition 9) betrachtet, sondern auf Basis der Anzahl der Elemente in der durch das Plugin identifizierten Menge konformer Traces T_{OK} die einfache Fitness f_{simple} (siehe Definition 7) berechnet. Ist $f_{simple} = 1$, so werden alle Traces des Ereignisprotokolls durch das Prozessmodell reflektiert.

Für Prozessmodell (a) ergibt sich dabei $f_{simple} = \frac{0}{500} = 0$. Das bedeutet, dass kein einziger der 500 Traces des Ereignisprotokolls vollständig durch dieses Modell ausgedrückt werden kann, was angesichts der unzusammenhängenden Modellstruktur auch zu erwarten ist. Das Modell reflektiert somit kaum die Eigenschaften des zugrundeliegenden Prozesses, der durch das Ereignisprotokoll dokumentiert ist, und ist daher nicht geeignet, Sicherheitsanforderungen, die sich auf kausale Beziehungen zwischen Aktivitäten, bzw. Aktivitätsreihenfolgen beziehen, zu überprüfen. Hinweise auf das simple Enthaltensein von Aktivitäten gemäß Anforderung **(A1a)** können allerdings dennoch entnommen werden.

Für das Prozessmodell (b) ergibt sich $f_{simple} = \frac{489}{500} = 0.978$ und für Prozessmodell (c) $f_{simple} = \frac{488}{500} = 0.976$. Dies zeigt, dass beide Prozessmodelle zwar die meisten, jedoch nicht alle Traces aus dem Ereignisprotokoll repräsentieren. Für den allgemeinen Fall bedeutet das, dass die beiden Prozessmodelle nicht für die Überprüfung von Sicherheitsanforderungen verwendet werden sollten, da die Gefahr besteht, dass sich unter diesen Traces auch abweichende Traces befinden und somit Abweichungen unerkannt bleiben. Da die im Ereignisprotokoll enthaltenen Abweichungen hier bekannt sind (siehe Tabelle 5.2), soll ein Blick auf die sich durch die Anwendung des Plugins `Conformance Checker` ergebende Menge T_{NOK} geworfen werden, um festzustellen, ob sich tatsächlich abweichende Traces unter den durch die jeweiligen Modelle nicht repräsentierten Traces befinden. Für Prozessmodell (b) ist $T_{NOK} = \{9, 18, 29, 30, 31, 34, 43, 47, 52, 57, 63\}$, für Prozessmodell (c) kommt zu ebendieser Menge noch Trace 5 hinzu[2]. Diese Traces entsprechen exakt den abweichenden Traces zur Sicherheitsanforderung **(A3)**,

[2]Dies ist nur der Fall, wenn das Prozessmodell (c) direkt nach der Erstellung durch das `Genetic algorithm plugin` in ein Petri-Netz konvertiert und der `Conformance Checker` ausgeführt wird. Wird das Petri-Netz als PNML-Datei abgespeichert und später erneut in ProM geladen, greift die bereits in Abschnitt 5.1.3 beschriebene Einschränkung, dass

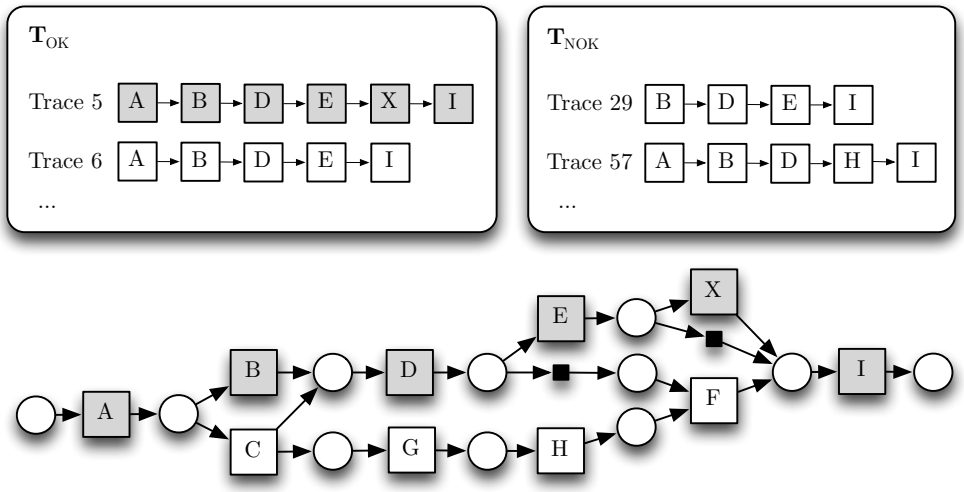

Abbildung 5.4.: Ausschnitt aus den Ergebnismengen T_{OK} und T_{NOK} der Petri-Netz-basierten Konformitätsprüfung mit dem Plugin `Conformance Checker` (oben) auf dem Prozessmodell (b) (unten). Der abweichende Trace 5 aus der Menge T_{OK} kann auf dem Prozessmodell erkannt werden, die abweichenden Traces 29 und 57 aus der Menge T_{NOK} jedoch nicht.

die nach Tabelle 5.2 im Ereignisprotokoll enthalten sind. Diese Abweichungen bleiben also bei Betrachtung der Prozessmodelle unerkannt, mit Ausnahme des abweichenden Trace 5, der von Modell (b) beschrieben wird, wie in Abb. 5.4 veranschaulicht. Zu sehen ist auch, dass die abweichenden Traces 29 und 57 in der Menge T_{NOK} nicht durch das Prozessmodell reflektiert und somit auf diesem nicht erkannt werden können.

Dies demonstriert nebenbei auf eindrucksvolle Weise, dass sowohl das Heuristic Mining als auch das Genetic Mining dazu in der Lage sind, selten auftretende Sequenzen (und das Genetic Mining auch selten auftretende Aktivitäten) bei der Modellkonstruktion auszuschließen. Bezerra u. Wainer [23] nutzen genau diese Eigenschaften aus, in dem sie die gezeigte Vorgehensweise mit initialer Modell-Extraktion und einer abschließenden Anwendung des `Conformance Checker` gezielt einsetzen, um Anomalien in Ereignisprotokollen zu identifizieren. Da die Abweichungen in diesem Fallbeispiel ausnahmslos in geringer Zahl enthalten sind, fallen sie in diese Kategorie. Treten die Abweichungen jedoch häufiger im

Ereignisse, die keine entsprechende Aktivität im Prozessmodell besitzen (Aktivität X) ignoriert werden, und der nicht-konforme Trace 5 wird nicht als solcher identifiziert.

Ereignisprotokoll auf, so sind sie auf diese Weise nicht mehr zu identifizieren, denn folglich werden sie aufgrund ihrer Häufigkeit nicht mehr als Noise deklariert und somit bei der Modellkonstruktion explizit berücksichtigt. Allerdings hat dies wiederum zur Folge, dass sie möglicherweise durch ein Prozessmodell ausgedrückt werden und somit bereits auf diesem erkannt werden können.

Um zu erkennen, ob bei der Suche nach Abweichungen auf den vorgestellten Prozessmodellen (a), (b) und (c) falsch positive Ergebnisse entstehen, werden die drei Modelle manuell daraufhin analysiert, welche Aktivitätssequenzen sich unter ihrer Verwendung generieren lassen. Vom vermuteten Startzustand $t1$ in Modell (a) geht keine Kante aus, die Netzstruktur ist darüberhinaus unzusammenhängend. Demnach ergibt sich aus diesem Modell keinerlei Aktivitätssequenz und folglich auch keine falsch positiven Ergebnisse. Die Analyse der Modelle (b) und (c) ergibt zunächst diejenigen Aktivitätssequenzen, die nach Tabelle 5.1 für den Prozess vorgesehen sind, Modell (b) beinhaltet zudem noch die Aktivitätssequenz $\langle A, B, D, E, X, I \rangle$, welche einer Abweichung entspricht. Darüber hinaus werden durch die Modelle keine weiteren Aktivitätssequenzen repräsentiert, so dass unter ihrer Verwendung ebenfalls keine falsch positiven Ergebnisse bei der Suche nach Abweichungen entstehen.

Durch Anwendung eines trivialen Verfahrens kann mit ProM 5.2 ein sogenanntes explizites und auch vollständiges Prozessmodell (d) (siehe Abb. 5.5) mit einer Fitness $f_{simple} = \frac{500}{500} = 1$ aus dem Ereignisprotokoll generiert werden. Auf diesem können alle Abweichungen zur Sicherheitsanforderung **(A3)** erkannt werden. Das Plugin `Explicit Model Miner` erstellt bei der Modellkonstruktion für jeden einzelnen im Ereignisprotokoll enthaltenen Trace einen eigenen Pfad von einer künstlichen, versteckten Initial-Transition des Modells zu einer ebensolchen Final-Transition. Wird der `Explicit Model Miner` direkt auf dem Ereignisprotokoll angewendet, entsteht demnach ein extrem großes Prozessmodell mit 500 Pfaden für die 500 enthaltenen Traces. Da viele Traces die gleiche Aktivitätssequenz enthalten, sind viele Pfade in mehrfacher Ausführung im Modell vorhanden. Soll die Häufigkeit von abweichenden Pfaden nicht betrachtet werden, so kann man dies umgehen, indem vor Anwendung des `Explicit Model Miner` ein Export des Ereignisprotokolls als `Grouped MXML Log (same sequences)` ausgeführt wird. Dadurch wird das Ereignisprotokoll derart reduziert, dass Traces mit gleicher Aktivitätssequenz nicht mehrfach auftreten. Dadurch sind im resultierenden Ereignisprotokoll nur noch 16 Traces enthalten. So viele Pfade besitzt auch das durch die anschließende Anwendung des `Explicit Model Miner` generierte Prozessmodell aus Abb. 5.5. Diese 16 Pfade repräsentieren exakt die vier konformen Traces des vorgegebenen Prozessmodells und die zusätzlichen 12 nicht-konformen Traces aus den Abweichungen zur Sicherheitsanforderung **(A3)** (siehe Tabelle 5.2). Auf diesem Prozessmodell

Tabelle 5.6.: Verfahren der Modell-Extraktion und Sicherheitsanforderungen: Anteil der identifizierten Abweichungen des Fallbeispiels Schadenersatzanspruch.

	(A1a)	**(A1b)**	**(A2a)**	**(A2b*)**	**(A3)**	**(A4)**
Prozessmodell (a) des α-Algorithmus (`Alpha algorithm plugin`)	1/1	0/1	0/1	-	1/12	-
Prozessmodell (b) des heuristischen Minings (`Heuristic miner`)	1/1	0/1	0/1	-	1/12	-
Prozessmodell (c) des genetischen Minings (`Genetic algorithm plugin`)	0/1	0/1	0/1	-	0/12	-
Explizites Prozessmodell (d) (`Explicit Model Miner`)	1/1	1/1	1/1	-	12/12	-

können daher nicht nur alle diese Abweichungen erkannt werden. Da das explizite Prozessmodell nicht übergeneralisiert, sind auch keine falsch positiven Ergebnisse bei der Suche nach Abweichungen möglich.

Zusammenfassung. Abschließend bietet Tabelle 5.10 einen Überblick über die Ergebnisse der indirekten Überprüfung der Sicherheitsanforderungen dieses Fallbeispiels mit den eingesetzten Verfahren. In diesem Abschnitt konnten zusammengefasst die folgenden wesentlichen Punkte demonstriert werden:

- Die Sicherheitsanforderungen **(A2b*)** und **(A4)** lassen sich grundsätzlich auf die hier vorgestellte Weise nicht überprüfen, da die nötigen Informationen nicht durch Petri-Netz-Prozessmodelle repräsentiert werden.
- Die *de facto* Prozessmodelle (a), (b) und (c) in Abb. 5.3 sind für die Überprüfung der Sicherheitsanforderungen des Fallbeispiels ungeeignet, denn Abweichungen bleiben zu einem Großteil unerkannt, wie in Tabelle 5.10 dargestellt.
- Es ist möglich mit einem trivialen Verfahren ein explizites Prozessmodell zu generieren, welches die Voraussetzungen dafür erfüllt, als Grundlage für die Überprüfung von Sicherheitsanforderungen eingesetzt zu werden (siehe Prozessmodell (d) in Abb. 5.5).
- Der `Conformance Checker` kann eingesetzt werden, um die Fitness f_{simple} zu berechnen. Wenn falsch negative Ergebnisse vermieden werden müssen, dürfen nur Prozessmodelle als Grundlage für die Überprüfung verwendet werden, für welche eine vollständige Fitness $f_{simple} = 1$ bezüglich

des Ereignisprotokolls erzielt wird. Andernfalls besteht die Gefahr, dass Abweichungen unerkannt bleiben.

- Sind Verfahren zur Modell-Extraktion dazu in der Lage, seltene Ereignisse oder Kausalitäten bei der Modellkonstruktion auszuschließen, können die resultierenden Modelle in Kombination mit der Petri-Netz-basierten Konformitätsprüfung eingesetzt werden, um Anomalien zu identifizieren. Dies betrifft auch diejenigen Abweichungen von Anforderungen, die die Eigenschaft besitzen, selten im Ereignisprotokoll aufzutreten. Dies wurde von Bezerra u. Wainer [23] bereits festgestellt.

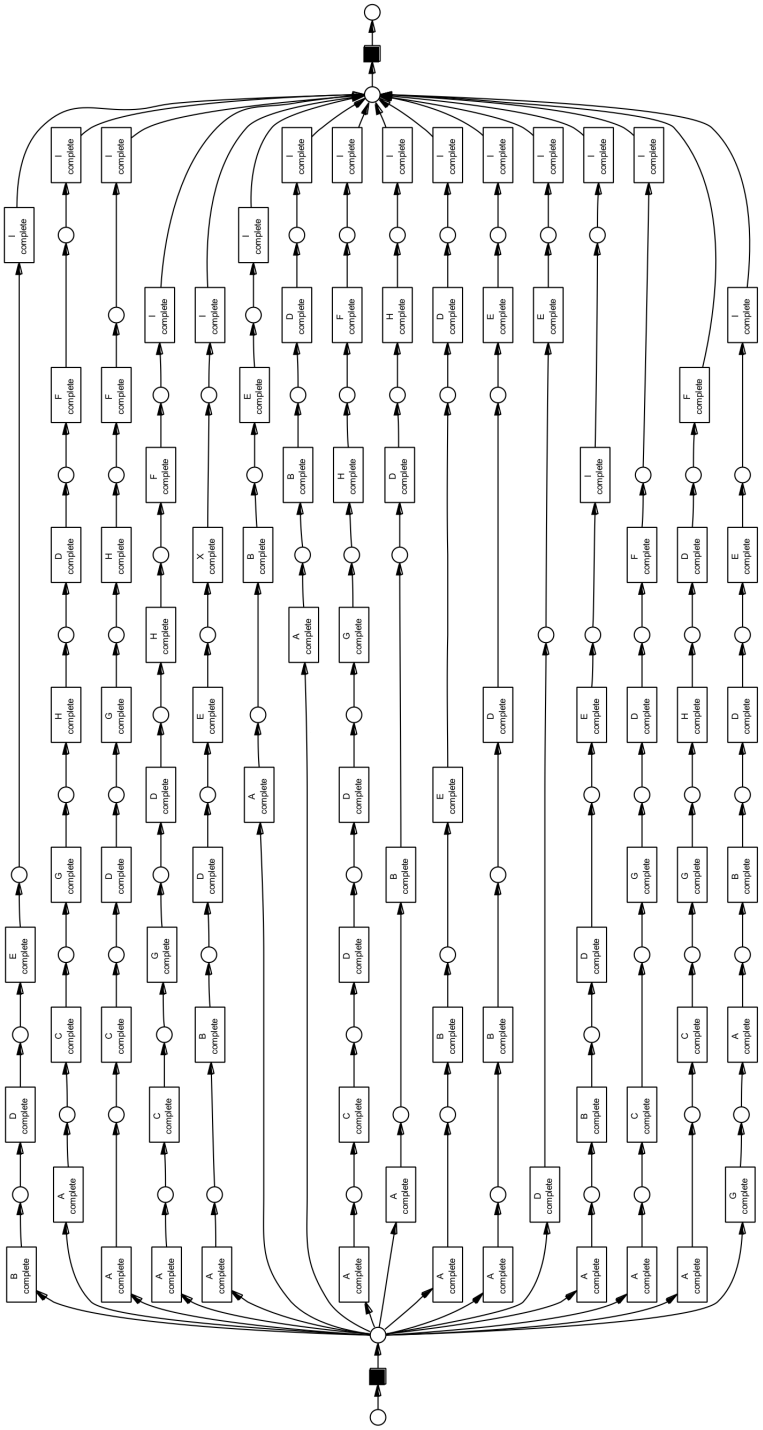

Abbildung 5.5.: Mit ProM 5.2 unter Verwendung des Plugins `Explicit model miner` aus dem Ereignisprotokoll generiertes explizites Prozessmodell (d).

5.2. Prozess Kreditvergabe

5.2.1. Beschreibung des Fallbeispiels

Dieses Fallbeispiel basiert auf dem Prozess zur Kreditvergabe einer Bank, der von Arsac et al. [18] beschrieben wird. Abb. 5.6 zeigt ein BPMN-Diagramm und das entsprechende Petri-Netz-Prozessmodell des Prozesses. Das Ziel dieses Prozesses

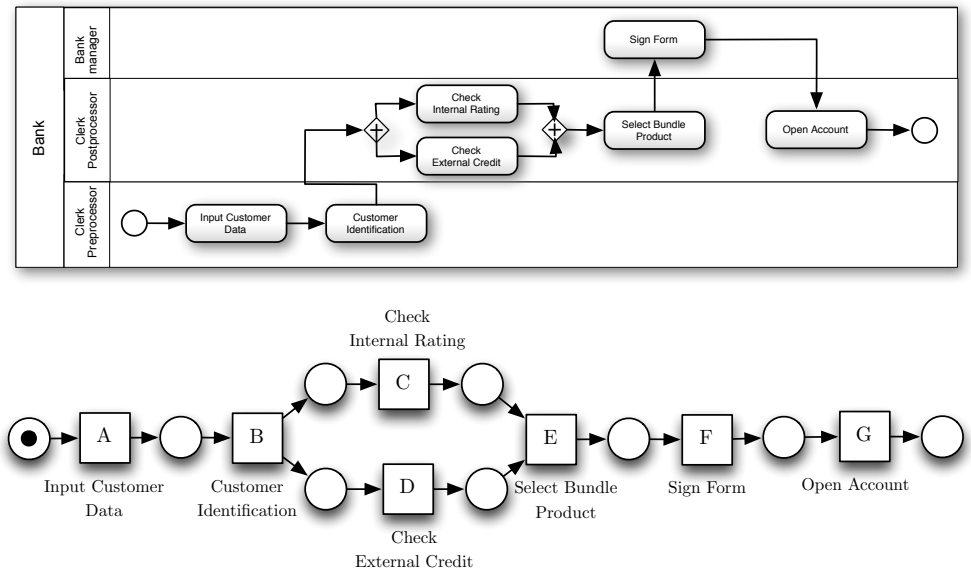

Abbildung 5.6.: Prozess zur Abwicklung von Kreditvergaben. Oben: BPMN-Diagramm aus [18], unten: Prozessmodell als Petri-Netz.

ist es, eine Kundenanfrage bezüglich einer Kreditvergabe zu überprüfen und zu genehmigen, was durch die in der Abbildung zu sehende Anordnung der Prozessaktivitäten erfolgt. Nach Aufnahme der Kundendaten und Identifikation des Kunden (A und B) werden Überprüfungen des Kunden angestellt (C und D). Das Ergebnis dieser Überprüfung dient als Basis für die automatisiert stattfindende Auswahl eines entsprechenden Produkts (E). Nach dem Einholen einer Unterschrift (F) wird das Konto eröffnet (G).

Diese Prozessaktivitäten werden entsprechend der sogenannten *Lanes* (dt. Bahnen) im BPMN-Diagramm drei verschiedenen Rollen innerhalb der Bank zugeordnet. Besitzt ein Teilnehmer im Prozess eine Rolle, bedeutet dies, dass er autorisiert ist, die der Rolle entsprechenden Aktivitäten auszuführen.

Tabelle 5.7.: RBAC-Vorgaben für das Fallbeispiel Kreditvergabe, Zuordnung von Rollen zu Teilnehmern und Aktivitäten.

Rolle	Teilnehmer	Aktivitäten
Bank manager	A_9, A_{10}, B_9, B_{10}	$A - G$
Clerk postprocessor	$A_5, A_6, A_7, A_8, B_5, B_6, B_7, B_8$	C, D, E, G
Clerk preprocessor	$A_1, A_2, A_3, A_4, B_1, B_2, B_3, B_4$	$A - B$

Für dieses Fallbeispiel werden zwei Abteilungen A und B mit je 10 Teilnehmern A_1–A_{10} bzw. B_1–B_{10} eingeführt. Die Zuordnung dieser Teilnehmer zu den Rollen des Prozesses wird durch die RBAC-Vorgaben in Tabelle 5.7 beschrieben. So gibt es in jeder Abteilung zwei Teilnehmer der Rolle *Bank manager* und je vier Teilnehmer der Rollen *Clerk postprocessor* und *Clerk preprocessor*. Die Zuordnung der Rollen zu den Aktivitäten des Prozesses wird entsprechend den Lanes des BPMN-Diagramms vorgenommen, wobei allerdings die Rolle *Bank manager* nicht nur Aktivität *Sign Form* (F), sondern auch alle anderen Prozessaktivitäten ausführen darf.

Desweiteren sollen in diesem Fallbeispiel alle Aktivitäten dieses Prozesses Eingabe/Ausgabe-Operationen repräsentieren, d.h. bei ihrer Ausführung wird jeweils ein Datenobjekt eingelesen (Eingabe) und ein Datenobjekt geschrieben (Ausgabe).

Die Prozessausführung durch die Teilnehmer in diesem Fallbeispiel wird durch den Einbau der in Abschnitt 2.3 erarbeiteten Sicherheitsanforderungen wie in Tabelle 5.8 beschrieben eingeschränkt.

5.2.2. Vorbereitung

Aufgrund der Komplexität dieses Fallbeispiels ist es nicht möglich, mit den Mitteln des Simulationswerkzeugs direkt ein Ereignisprotokoll zu erzeugen, welches alle der hier vorgegebenen Sicherheitsanforderungen sinngemäß umsetzt. Auf welche Weise dennoch sichergestellt werden kann, dass das für die spätere Überprüfung der Sicherheitsanforderungen verwendete Ereignisprotokoll vor Einbau der Abweichungen konform ist, wird im Folgenden beschrieben:

- Die RBAC-Vorgaben aus Tabelle 5.7 können über Simulationsparameter gesteuert werden, so dass die Einhaltung der Sicherheitsanforderungen **(A6a)** und **(A6b)** durch die Simulation gegeben ist. Dabei kann allerdings die Anforderung **(A7a)** verletzt werden, wenn ein Teilnehmer der Rolle *Bank manager*, der die Aktivität *Sign Form* (F) ausführt, in der gleichen Instanz auch die Aktivität *Select Bundle Product* (E) ausführt, da er

Tabelle 5.8.: Sicherheitsanforderungen zum Fallbeispiel Kreditvergabe.

Name	Beschreibung
(A6a)	Aktivität *Sign Form* (F) darf nur von den Teilnehmern A_9, A_{10}, B_9, B_{10} ausgeführt werden.
(A6b)	Alle Teilnehmer müssen die Berechtigungen der RBAC-Vorgaben in Tabelle 5.7 einhalten.
(A7a)	Führt Teilnehmer $T1$ die Aktivität *Select Bundle Product* (F) aus, so muss die Aktivität *Sign Form* (F) in der *gleichen Instanz* des Prozesses von einem Teilnehmer $T2$ ausgeführt werden, so dass $T1 \neq T2$ gilt.
(A7b)	Wird Aktivität *Input Customer Data* (D) für einen Kunden K_1 ($\#_{Kunde} = K_1$) ausgeführt, so muss die gleiche Aktivität in einer *anderen Instanz* des Prozesses für einen Kunden K_2 ausgeführt werden, so dass $K1 \neq K2$ gilt. Somit wird gefordert, dass jeder Kunde nur eine Kreditvergabe beantragen darf.
(A8)	Aktivität *Customer Identification* (B) mit Zugriff auf Daten der Stufe *secret* ($\#_{Stufe} = secret$) darf nur von Teilnehmer A_1 ausgeführt werden.
(A9)	Alle Teilnehmer, die an der Ausführung einer Prozessinstanz beteiligt sind, müssen zur gleichen Partei gehören.
(A10)	Keine Prozessaktivität darf bei der Ausführung durch einen Teilnehmer der Abteilung A Daten als Eingabe verwenden, die von einem Teilnehmer der Abteilung B bei einer beliebigen Prozessaktivität des Prozesses als Ausgabe erzeugt wurden. Für zwei Ereignisse e_1 und e_2 mit $\#_{Abteilung}(e_1) \neq \#_{Abteilung}(e_2)$ muss also gelten: $\#_{Eingabe}(e_1) \neq \#_{Ausgabe}(e_2)$ und vice versa.
(A11)	Die Teilnehmer am Prozess dürfen nicht ableiten können, wie die Ausgaben der Aktivitäten *Check Internal Rating* (C) und *Check External Credit* (D) die Ausgaben der Aktivität Select Bundle Product (E) beeinflussen.

dies nach den RBAC-Vorgaben darf. Um dies zu verhindern, werden für die Rolle *Bank manager* die Simulationsparameter derart gewählt, dass Teilnehmer der Rolle *Bank manager* während der Simulation nur die Aktivität *Sign Form* (F) ausführen, anstatt ihr volles Handlungspotenzial auszuschöpfen.

- Damit gemäß Sicherheitsanforderung **(A7b)** sichergestellt ist, dass keine Instanz den gleichen Kunden behandelt, wird dem Ereignisattribut $\#_{Kunde}$ während der Simulation bei jeder neuen Instanz bzw. für jeden Trace ein um 1 inkrementierter Wert zugewiesen. Zur Einhaltung der Sicherheitsanforderung **(A8)** wird das Ereignisattribut $\#_{Stufe}$ durch die Simulation auf den konstanten Wert *public* gesetzt.
- Zur Benennung der Datenobjekte, die durch die Prozessaktivitäten gelesen und geschrieben werden, wird den Ereignisattributen $\#_{Eingabe}$ so-

Tabelle 5.9.: Abweichungen im Ereignisprotokoll zum Fallbeispiel Kreditvergabe.

Trace	Abweichende Eigenschaften	Verletzt Sicherheitsanforderung(en)
5	$\#_{Teilnehmer}(F) = A_7$	**(A6a)**, da Aktivität F von Teilnehmer A_7 ausgeführt wird, **(A6b)** außerdem
7	$\#_{Teilnehmer}(A) = A_7$	**(A6b)**, da entgegen der RBAC-Vorgaben
8	$\#_{Teilnehmer}(C) = A_2$	**(A6b)**, da entgegen der RBAC-Vorgaben
9	$\#_{Teilnehmer}(E) = A_1$	**(A6b)**, da entgegen der RBAC-Vorgaben
22	$\#_{Teilnehmer}(F) = A_{10}$, $\#_{Teilnehmer}(E) = A_{10}$	**(A7a)**, da Teilnehmer A_{10} sowohl Aktivität F als auch Aktivität E ausführt
54,55	$\sigma(54) : \#_{Kunde}(A) = A_{54}$, $\sigma(55) : \#_{Kunde}(A) = A_{54}$	**(A7b)**, da die Instanzen 54 und 55 den gleichen Kunden behandeln
60	$\#_{Stufe}(A) = secret$, $\#_{Teilnehmer}(A) = A_4$	**(A8)**, da Teilnehmer A_4 auf Daten der Stufe *secret* zugreift
65	$\#_{Teilnehmer}(F) = B_9$, $\#_{Abteilung}(F) = B$	**(A9)** , da Teilnehmer B_9 eine Aktivität in einer Instanz der Abteilung A ausführt
100, 101	$\sigma(100) : \#_{Ausgabe}(C) = A_{3823}$, $\sigma(101) : \#_{Eingabe}(B) = A_{3823}$	**(A10)**, da in Instanz 101 von Abteilung B ein in Instanz 100 von Abteilung A erzeugtes Datenobjekt gelesen wird

wie $\#_{Ausgabe}$ ein zufälliger Wert zwischen 1 und 20000 zugewiesen. Zur Durchsetzung von Sicherheitsanforderungen **(A9)** sowie **(A10)** werden für die zwei Abteilungen A und B des Fallbeispiels getrennte Simulationen ausgeführt und die resultierenden Ereignisprotokolle anschließend zusammengeführt. Dazu werden für Abteilung A 100 Traces simuliert, für Abteilung B 200 Traces. Die ersten 100 Traces des Ereignisprotokolls für B werden dann durch die 100 Traces der Abteilung A ersetzt, so dass keine Überschneidungen der IDs auftreten können. Damit sichergestellt ist, dass die beiden Abteilungen unterschiedliche Kunden und Datenobjekte bearbeiten, wird bei der Wertezuweisung der entsprechenden Attribute jeweils ein String $A\_$ bzw. $B\_$ vorangestellt.

In das nach den obigen Angaben erzeugte Ereignisprotokoll werden anschließend Abweichungen zu den einzelnen Sicherheitsanforderungen eingebaut, wie sie in Tabelle 5.9 beschrieben sind. Lediglich für die Sicherheitsanforderung **(A11)** wird auf den Einbau einer Abweichung verzichtet. Der implizite Informationsfluss von den Ausgaben der Aktivitäten *Check Internal Rating* (C) und *Check External Credit* (D) zu den Ausgaben der Aktivität *Select Bundle Product* (E) ergibt sich direkt aus dem hier vorgestellten Prozessablauf und kann ohne Einführung

weiterer Einschränkungen nicht verhindert werden[3]. Somit ist es unumgänglich, dass Teilnehmer, die diese Aktivitäten im Rahmen der Prozessausführung anwenden, Kenntnisse über die zugrundeliegende Berechnung mit der Aktivität *Select Bundle Product* (E) gewinnen.

5.2.3. Überprüfung der Sicherheitsanforderungen

Soweit möglich, wird in diesem Abschnitt das Verfahren der Log-basierten Verifikation zur direkten Überprüfung von Sicherheitsanforderungen auf Grundlage des vorbereiteten Ereignisprotokolls angewendet. Ebenso wird demonstriert, wie für die geeigneten Sicherheitsanforderungen **(A6b)** sowie **(A9)** eine indirekte Überprüfung auf extrahierten Modellen der Organisationsperspektive (Abschnitt 3.3.2) aussehen kann.

(A6a) und (A6b). Für die Prüfung der beiden Sicherheitsanforderungen bezüglich Zugriffskontrollrechten muss für jede Aktivität kontrolliert werden, ob sie nur von Teilnehmern ausgeführt wird, die entsprechend autorisiert sind. Das Listing in Anhang C.2 enthält eine Formel für die Überprüfung von **(A6a)** unter Verwendung des Plugins `LTL Checker`. Mit dieser Formel kann der von **(A6a)** abweichende Trace 5 identifiziert werden. Die manuelle Entwicklung von LTL-Formeln zur vollständigen Überprüfung aller Zugriffskontrollrechte gemäß **(A6b)** ist eine mühsame Aufgabe, die durch den Ansatz von Baumgrass et al. [21] unterstützt werden kann. Es genügt die Angabe eines RBAC-Modells im XML-Format. Dabei werden das Prozessmodell, Rollen und Teilnehmer des Prozesses sowie entsprechende Zuordnungen spezifiziert. Das RBAC-Modell für das vorliegende Fallbeispiel ist in Anhang C.3 aufgeführt. Zur Durchführung der Überprüfung wird die XML-Datei des RBAC-Modells unter Verwendung des Import-Plugins `RBAC model` in ProM geladen. Anschließend wird das RBAC-Modell mit Hilfe des Plugins `RBAC to LTL conversion` in ein LTL-Modell bestehend aus einer Menge von LTL-Formeln transformiert. Dieses LTL-Modell kann nun zusammen mit dem Ereignisprotokoll in das Plugin `LTL Checker` geladen werden. Dabei können die automatisiert erstellten LTL-Formeln zur Prüfung der Zugriffskontrollrechte einzeln oder gleichzeitig überprüft werden. Bei der gleichzeitigen Überprüfung werden alle Abweichungen zu **(A6b)**, nämlich die Traces 5, 7, 8 und 9 aus Tabelle 5.9 identifiziert.

[3] Dies wäre z. B. durch eine Aufteilung der Rolle *Clerk Postprocessor* in zwei getrennte Gruppen von Teilnehmern möglich, so dass Teilnehmer der ersten Gruppe nur die Eingaben der Berechnung und Teilnehmer der zweiten Gruppe nur die Ausgaben der Berechnung sehen können.

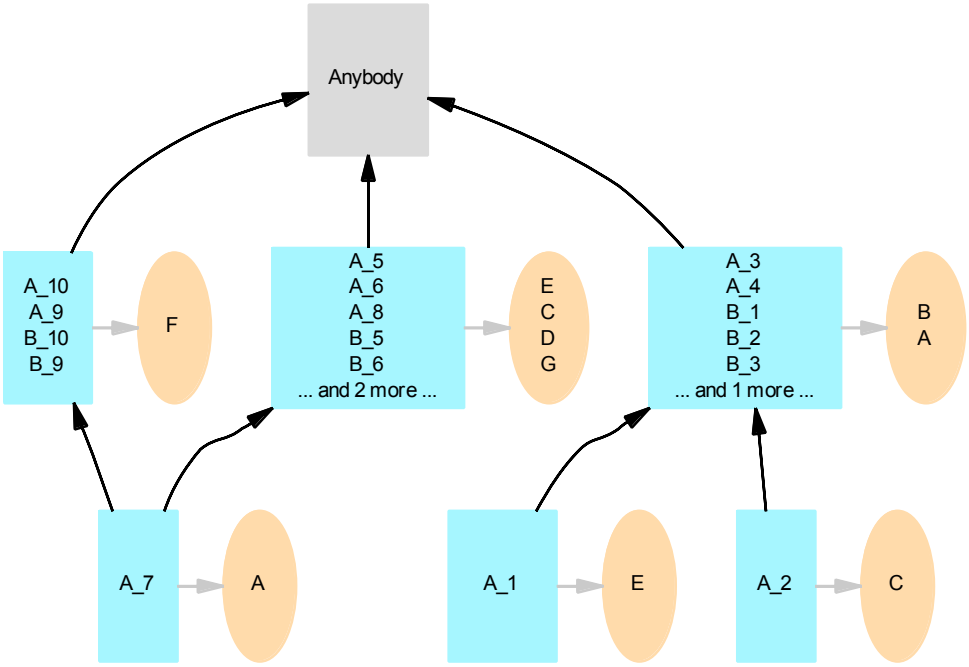

Abbildung 5.7.: Hierarchisches Rollenmodell; Ergebnis der Anwendung des Plugins `Role Hierarchy Miner` (ProM 5.2) auf dem Ereignisprotokoll des Fallbeispiels.

Abweichungen von den RBAC-Vorgaben können auch auf dem Rollenmodell in Abbildung 5.7 erkannt werden, welches durch Anwendung des Plugins `Role Hierarchy Miner` aus dem Ereignisprotokoll extrahiert wird. In der mittleren Ebene dieses Modells sind die drei Rollen des Prozesses mit ihren entsprechenden Teilnehmern und Aktivitäten zu sehen. Da das Modell hierarchisch von unten (mehr Rechte) nach oben (weniger Rechte) aufgebaut ist, zeigt die unterste Ebene die vier Abweichungen von **(A6b)**: Teilnehmer A_7 ist einmal Aktivität A (Trace 7) und einmal Aktivität F (Trace 5) zugeordnet, für die Teilnehmer A_1 und A_2 sind Zuordnungen zu den Aktivitäten E (Trace 9) bzw. C (Trace 8) durch die Pfeile im Modell angegeben, woraus sich ergibt, dass die Teilnehmer die entsprechenden Aktivitäten dem Ereignisprotokoll nach mindestens einmal ausgeführt haben.

(A7a) und (A7b). Eine LTL-Formel für die Überprüfung der Sicherheitsanforderung **(A7a)** zur Funktionstrennung ist im Listing in Anhang C.2 enthalten.

Mit dieser Formel kann unter Verwendung des Plugins `LTL Checker` der von **(A7a)** abweichende Trace 22 identifiziert werden. Darüber hinaus können Sicherheitsanforderungen zur Funktionstrennung auch mit Hilfe des weiter oben beschriebenen RBAC-Modells spezifiziert werden. Dazu wird für Aktivität F in Zeile 36 des Listings in Anhang C.3 mit dem Tag `<dynamicExclusion>` festgelegt, dass Aktivität E in der gleichen Prozessinstanz nicht vom gleichen Teilnehmer ausgeführt werden darf. Durch die Transformierung des RBAC-Modells mit dem Plugin `RBAC to LTL conversion` wird ebenfalls eine entsprechende LTL-Formel für die Prüfung von **(A7a)** generiert.

Zur Überprüfung der Sicherheitsanforderung **(A7b)** ist es erforderlich, das Ereignisattribut $\#_{kunde}$ instanzübergreifend zu vergleichen. Da die instanzübergreifende Betrachtung mit dem Plugin `LTL Checker` nicht möglich ist, müssen zuvor alle Instanzen des Ereignisprotokolls zu einer einzelnen Instanz zusammengesetzt werden[4]. Dazu genügt es, in einem ersten Schritt alle öffnenden und schließenden Tags für Prozessinstanzen (`<ProcessInstance>` bzw. `</ProcessInstance>`) aus dem Ereignisprotokoll zu entfernen und in einem zweiten Schritt einen öffnenden Tag vor das erste Ereignis und einen schließenden Tag nach dem letzten Ereignis zu plazieren. Anschließend kann auf diesem Ereignisprotokoll eine gewöhnliche LTL-Formel für die Überprüfung von **(A7b)** (siehe Listing C.3) mit dem Plugin `LTL Checker` angewendet werden. Da das Ereignisprotokoll nur noch aus einem einzelnen Trace besteht, für welchen die LTL-Formel entweder wahr oder falsch ist, kann zwar erkannt werden, ob eine Abweichung von **(A7b)** vorliegt oder nicht, es ist aber nicht möglich, zu erkennen, wieviele Abweichungen im Ereignisprotokoll enthalten sind, oder welche (ursprünglichen) Instanzen von der Abweichung betroffen sind.

(A8). Mit der Sicherheitsanforderung **(A6a)** wurde die Zuordnung von Teilnehmern zu Aktvititäten festgelegt. Die Sicherheitsanforderung **(A8)** erweitert dies um eine Zuordnung von Teilnehmern zu Aktivitäten und Datenobjekten. Die Erstellung einer entsprechenden LTL-Formel, die mit dem Plugin `LTL Checker` geprüft werden kann, ist trivial und wird in Listing C.2 vorgestellt.

(A9). Diese Sicherheitsanforderung schreibt vor, dass innerhalb einer Instanz nur Teilnehmer der gleichen Abteilung Aktivitäten ausführen dürfen. Dafür lässt sich eine entsprechende LTL-Formel erstellen, die mit dem Plugin `LTL Checker` überprüft werden kann (siehe Listing C.2). Es ist darüber hinaus möglich, die Arbeitstrennung zwischen den beiden Abteilungen auf einem sozialen Netzwerk

[4] Dies wird auch von Baumgrass et al. [21] vorgeschlagen.

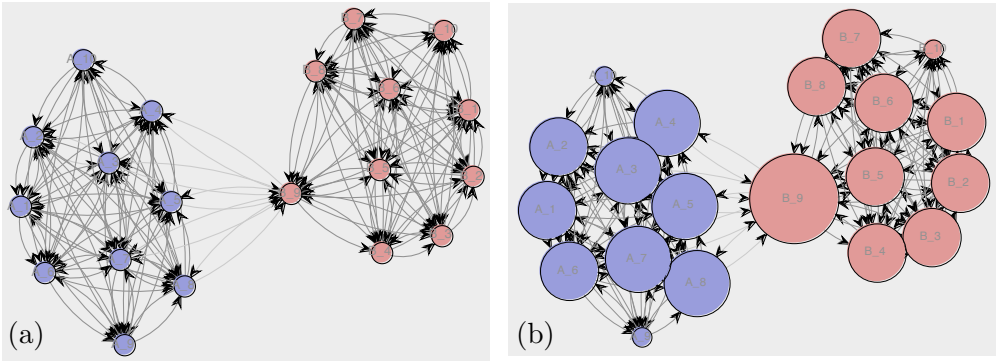

Abbildung 5.8.: Soziales Netzwerk; Ergebnis der Anwendung des Plugins `Mine for a Working-Together Social Network` (ProM 6.1) auf dem Ereignisprotokoll des Fallbeispiels. In Netzwerk (b) ist die Größe der Knoten relativ zur Anzahl ihrer Kanten dargestellt.

zu überprüfen, welches durch die Anwendung des Plugins `Mine for a Working-Together Social Network` extrahiert werden kann. Das Plugin erstellt ein soziales Netzwerk auf Grundlage der in Abschnitt 3.3.2 beschriebenen Metrik zur Zusammenarbeit, so dass die Beziehungen (Kanten) zwischen Teilnehmern (Knoten) dadurch definiert sind, dass gemeinsam an einem Fall (also in einer Instanz) gearbeitet wird. Genau solche Beziehungen sind nach **(A9)** nur zwischen Teilnehmern einer Abteilung gestattet. In dem entsprechenden sozialen Netzwerk in Abb. 5.8 ist aufgrund der Abweichung in Trace 65 (siehe Tabelle 5.9) eine solche Beziehung auch zwischen dem Knoten des Teilnehmers B_9 der Abteilung B und verschiedenen Teilnehmern der Abteilung A eingetragen. Somit lässt sich die Abweichung von der Sicherheitsanforderung **(A9)** auf dem sozialen Netzwerk erkennen. Es ist auch ersichtlich, welcher Teilnehmer (hier B_9, je nach Blickwinkel aber auch die beteiligten Teilnehmer der Abteilung A) die Sicherheitsanforderung verletzt. Allerdings kann auf Grundlage des sozialen Netzwerks nicht angegeben werden, welche Instanzen von der Abweichung betroffen sind.

(A10). Auch die Überprüfung der Sicherheitsanforderung zum expliziten Informationsfluss macht eine instanzübergreifende Betrachtung der Ereignisattribute erforderlich. Wird ein Datenobjekt über ein beliebiges Ereignis in einer Instanz der Abteilung A geschrieben und anschließend über ein beliebiges Ereignis einer Instanz der Abteilung B gelesen, so findet ein expliziter Informationsfluss von Abteilung A zu Abteilung B statt. Werden wie zuvor für Sicherheitsanforderung **(A7b)** beschrieben alle Instanzen des Ereignisprotokolls zu einer einzelnen In-

stanz zusammengeführt, dann kann eine LTL-Formel erstellt werden, welche mit dem Plugin `LTL Checker` die Datenfluss-Eigenschaft erfolgreich überprüft (siehe Listing C.3). Es gilt allerdings auch hier, dass durch die Zusammenführung der Instanzen bei der Überprüfung mit dem Plugin `LTL Checker` nicht mehr erkannt werden kann, welche Instanzen im Einzelnen an der Abweichung beteiligt sind.

(A11). Für die Sicherheitsanforderung bezüglich des impliziten Informationsflusses wurden wie in Abschnitt 5.2.2 beschrieben keine Abweichungen im Ereignisprotokoll eingebaut, da das Bestehen des impliziten Informationsflusses von den Ausgaben der Aktivitäten *Check Internal Rating* (C) und *Check External Credit* (D) zu den Ausgaben der Aktivität *Select Bundle Product* (E) sich direkt aus dem vorgestellten Prozessablauf ergibt. Stattdessen soll an dieser Stelle diskutiert werden, ob eine *missbräuchliche Ausnutzung des impliziten Informationsflusses* auf Grundlage des durch das Ereignisprotokoll dokumentierten Prozessablaufs aufgedeckt werden kann. Dazu muss die Eigenschaft der missbräuchlichen Ausnutzung definiert werden. Eine solche könnte z. B. vorliegen, wenn die drei genannten Aktivitäten in einer Prozessinstanz wiederholt ausgeführt werden, so dass aus dem Vergleich von Eingaben und Ausgaben der Berechnung mit der Aktivität *Select Bundle Product* (E) Rückschlüsse auf die Berechnung selbst gezogen werden können. Die nicht erlaubte wiederholte Ausführung dieser Aktivitäten entspricht dann einer strukturellen Sicherheitsanforderung über die Aktivitätsreihenfolge, wie sie für das Fallbeispiel im vorigen Abschnitt 5.1 betrachtet wurden. Sie kann z. B. mit einer entsprechenden LTL-Formel mit Hilfe der Log-basierten Verifikation mit dem Plugin `LTL Checker`, oder auch anhand des Petri-Netz-Prozessmodells in Abb. 5.6 mit dem Plugin `Conformance Checker` überprüft werden. Denkbar sind weitere Eigenschaften einer missbräuchlichen Ausführung, wie z. B. die Existenz verschiedener Prozessinstanzen, die jeweils nur bis zur Aktivität *Select Bundle Product* (E) gelangen und dann nicht fortgeführt werden. Solange diese Eigenschaften auf Grundlage der durch das Ereignisprotokoll ausgedrückten Fakten entscheidbar sind, kann ein Verfahren gefunden werden, welches die Überprüfung durchführt.

Zusammenfassung. Abschließend bietet Tabelle 5.10 einen Überblick über das Ergebnis der Überprüfung der berücksichtigten Sicherheitsanforderungen des Fallbeispiels mit den in diesem Abschnitt betrachteten Verfahren. Daraus ergeben sich die folgenden Erkenntnisse:

- Mit dem Verfahren der Log-basierten Verifikation können alle betrachteten Sicherheitsanforderungen erfolgreich überprüft werden. Allerdings erfordert die Überprüfung von Inter-Instanz- bzw. Hypereigenschaften,

Tabelle 5.10.: Process Mining-Verfahren und Sicherheitsanforderungen: Anteil der identifizierten Abweichungen des Fallbeispiels Kreditvergabe.

	(A6a)	**(A6b)**	**(A7a)**	**(A7b)**	**(A8)**	**(A9)**	**(A10)**
Log-basierte Verifikation (`LTL Checker`)	1/1	4/4	1/1	1/1*	1/1	1/1	1/1*
Log-basierte Verifikation (via `RBAC to LTL conv.`)	1/1	4/4	1/1	-	-	-	-
Modell-Extraktion (`Role Hierarchy Miner`)	1/1	4/4	-	-	-	-	-
Modell-Extraktion (`Mine f. a. Working-Together Social Netw.`)	-	-	-	-	-	1/1	-

* Nur bei Zusammenführung aller Instanzen des Ereignisprotokolls zu einer einzelnen Instanz.

wie sie den Sicherheitsanforderungen (A7b) und (A10) zugrundeliegen, die Zusammenführung aller Instanzen des Ereignisprotokolls zu einer einzelnen Instanz. Dies kann durch eine Erweiterung der Verfahren behoben werden.

- Aus den Ergebnismengen T_{OK} bzw. T_{NOK} der Log-basierten Verifikation geht nur hervor, ob ein Trace eine Sicherheitsanforderung erfüllt, oder nicht. Welche Elemente (z. B. welcher Teilnehmer oder welche Datenobjekte) eines abweichenden Traces aus T_{NOK} für die Klassifikation eines Traces als nicht-konform verantwortlich sind, wird durch die gegenwärtige Implementierung des Verfahrens nicht visualisiert. Hier besteht ebenfalls Verbesserungspotenzial.
- Der Ansatz von Baumgrass et al. [21] ist bei der Überprüfung von RBAC-Vorgaben sehr hilfreich, da der manuelle Aufwand für die Erstellung einer Reihe entsprechender LTL-Formeln entfällt.
- Abweichungen können auch auf bestimmten Modellen der organisatorischen Perspektive erkannt werden. Dies wurde in diesem Abschnitt für die Sicherheitsanforderungen **(A6a)** bzw. **(A6b)** sowie **(A9)** gezeigt.

6. Diskussion

Im ersten Abschnitt 6.1 werden die Ergebnisse der in Kapitel 5 vorgestellten Fallstudien diskutiert. Anschließend wird die Validität der Fallstudien in Abschnitt 6.2 thematisiert. Der letzte Abschnitt 6.3 ist einer Einordnung dieser Arbeit in den Kontext verwandter Arbeiten gewidmet.

6.1. Ergebnisse der Fallstudien

Mit Hilfe der in Kapitel 5 durchgeführten Fallstudien konnte gezeigt werden, dass sich alle hier betrachteten Sicherheitsanforderungen, die sich auf Safety-Eigenschaften beziehen, mit den Methoden der Log-basierten Verifikation überprüfen lassen. Dies ist der Fall, da es sich bei Safety-Eigenschaften per Definition um Intra-Instanz-Eigenschaften handelt, die durch die Betrachtung einzelner Traces mit den bestehenden Verfahren zur Log-basierten Verifikation entschieden werden können. Diese erlauben jedoch nicht die gleichzeitige Betrachtung der Eigenschaften mehrerer Traces, so dass sich Inter-Instanz-Eigenschaften bzw. Hypereigenschaften, wie die Sicherheitsanforderungen **(A7b)** und **(A10)**, durch die gewöhnliche Anwendung der Verfahren nicht entscheiden lassen. Es wurde in dieser Arbeit gezeigt, dass durch die Zusammenführung aller Instanzen des Ereignisprotokolls zu einer einzelnen Instanz die Einschränkung der bestehenden Verfahren umgangen werden kann und die genannten Sicherheitsanforderungen auf diese Weise überprüft werden können. Diese Vorgehensweise ist allerdings lediglich als eine provisorische Lösung zu betrachten, da durch die Zusammenführung auch Informationen verloren gehen. So kann z. B. nicht erkannt werden, welche Traces für Abweichungen von zu prüfenden Eigenschaften betroffen sind. Eine Erweiterung von Verfahren der Log-basierten Verifikation um die Formulierung von Eigenschaften über mehrere Instanzen ist daher nicht nur wünschenswert, sondern schlichtweg notwendig, um auch Inter-Instanz-Sicherheitsanforderungen bei der Überprüfung abdecken zu können. Zudem ist es sinnvoll, die Ergebnisse der Verifikation nicht auf die Trace-Ebene zu beschränken. Wird mit Hilfe der bestehenden Verfahren ein Trace auf Grundlage einer komplexen, d. h. aus mehr als einer einzelnen Element-Eigenschaft zusammengesetzten Formel als nicht-konform klassifiziert, so ist es mit manuellem Aufwand verbunden herauszufinden, welche Elemente des Traces die eigentliche

Abweichung ausmachen. Daher gilt Folgendes: können Eigenschaften über alle Elemente eines prozessorientierten Ereignisprotokolls festgelegt und verifiziert werden, sollte das Ergebnis der Verifikation die individuelle Lokalisierung von nicht-konformen Elementen berücksichtigen. Demnach kann die Betrachtung weiterer Ebenen des Ereignisprotokolls im Ergebnis der Verifikation neben abweichenden Traces auch z. B. abweichende Ereignisse oder abweichende Attribute berücksichtigen. Sobald also auch (i) Inter-Instanz- bzw. Hypereigenschaften formuliert und mit dem durch das Ereignisprotokoll dokumentierte Verhalten verglichen werden können und (ii) eine Klassifikation in konforme bzw. nicht-konforme Elemente des Ereignisprotokolls nicht nur auf Trace-Ebene möglich ist, stellt die Log-basierte Verifikation ein noch mächtigeres Werkzeug für die Compliance-Prüfung als zuvor dar.

Angesichts der gerade beschriebenen Mächtigkeit der Verfahren der Log-basierten Verifikation stellt sich im Anschluss die Frage, welchen Beitrag Verfahren der Petri-Netz-basierten Konformitätsprüfung und der Modell-Extraktion für die Überprüfung von Sicherheitsanforderungen darüber hinaus leisten können. Die Anwendung der Petri-Netz-basierten Konformitätsprüfung in der ersten Fallstudie demonstriert, dass mit Hilfe des Replays eine Abweichung vom vorgesehenen Kontrollfluss (i) auf Grundlage der markenbasierten Fitness f individuell bewertet werden und zudem (ii) die Art der Abweichung genauer spezifiziert werden kann (z. B. Fehlen einer bestimmten Prozessaktivität). Allerdings ist der Einsatz der Petri-Netz-basierten Konformitätsprüfung zur Prüfung von Kontrollfluss-Eigenschaften nur dann zweckmäßig, wenn tatsächlich ein *de jure* Prozessmodell existiert, bzw. der Prozessablauf strukturiert ist und vollständig durch ein Petri-Netz definiert werden kann. Dadurch werden nicht nur flexible Prozesse grundsätzlich von einer auf Replay basierten Konformitätsprüfung ausgeschlossen. Es ist zudem nicht möglich, partielle Kontrollfluss-Eigenschaften wie sie durch die Sicherheitsanforderungen **(A1a)** - **(A2a)** definiert werden, individuell überprüfen zu lassen. Dazu ist es weiterhin unerlässlich, Verfahren der Log-basierten Verifikation auch für die Überprüfung von Kontrollfluss-Eigenschaften einzusetzen. Auch die Überprüfung von Kontrollfluss-Eigenschaften bezüglich deklarativer Prozessmodelle, die zur Spezifikation von vergleichsweise flexiblen Prozessen eingesetzt werden können (z. B. Declare [7] oder ConDec [58]), wird mit Hilfe von Verfahren der Log-basierten Verifikation abgedeckt.

Auch die in dieser Arbeit angewendeten Verfahren der Modell-Extraktion spielen für die Überprüfung von Sicherheitsanforderungen eine untergeordnete Rolle. Im besten Fall ermöglichen sie eine explorative Herangehensweise, indem sie eine kompakte Repräsentation und Visualisierung bestimmter Elemente des Ereignisprotokolls bieten. Dadurch können Abweichungen bei manueller Betrachtung erkannt werden, ohne dass wie bei der Log-basierten Verifikation

zuvor eine Spezifikation von zu überprüfenden Eigenschaften vorgenommen werden muss. Allerdings liefert die Modell-Extraktion unter Umständen (i) kein kompaktes oder sinnvolles Ergebnis und (ii) können Abweichungen unerkannt bleiben, wenn sie durch das extrahierte Modell nicht repräsentiert werden. Daher ist beim Einsatz von konventionellen Verfahren der Modell-Extraktion besondere Vorsicht angebracht.

Um die Nützlichkeit der Modell-Extraktion für die Überprüfung von Sicherheitsanforderungen zu verbessern, müssen vermehrt Verfahren entwickelt werden, die Abweichungen nicht als Noise deklarieren und entsprechend ignorieren, sondern diese bei der Modellkonstruktion explizit berücksichtigen. Darüber hinaus kann die Modell-Extraktion, wie bei Accorsi u. Stocker [12] vorgeschlagen, auch eine besondere Stellung im Zusammenhang mit der Überprüfung von Informationsflüssen einnehmen, wenn Verfahren entwickelt werden, die Informationsflussmodelle aus Ereignisprotokollen extrahieren. Die Extraktion von Modellen aus einer Informationsfluss- bzw. Datenflussperspektive wurde im Kontext des Process Minings bisher vernachlässigt. Dabei können Informationsflussmodelle als Grundlage für die Überprüfung der Hypereigenschaften, die Informationsfluss-Richtlinien zugrundeliegen, verwendet werden. Mit RecIF wurde die Durchführbarkeit eines derartigen Ansatzes zur Erkennung von expliziten Informationsflüssen bereits von Accorsi et al. [15] demonstriert. Die Extraktion von Informationsflussmodellen im IFnet-Dialekt auf Basis von gefärbten Petri-Netzen (*colored petri net*, CPN) [13] aus Ereignisprotokollen wird von Stocker [74] derzeit bearbeitet. Ansätze in dieser Richtung beschränken sich jedoch auf die Überprüfung von Sicherheitsanforderungen zur Informationsflusskontrolle und bieten somit wenig Flexibilität im Hinblick auf andere zu untersuchende Eigenschaften. Aus diesem Grund ist die weiter oben vorgeschlagene Weiterentwicklung von Ansätzen der Log-basierten Verifikation um die generelle Eignung zur Überprüfung von Hypereigenschaften unumgänglich, sollen auch Intra-Instanz Sicherheitsanforderungen wie **(A7b)** ohne Bezug zur Informationsflusskontrolle in die Überprüfung eingeschlossen werden.

6.2. Validität der Fallstudien

Nach Runeson u. Höst [68] muss nach Durchführung einer Fallstudie die Validität dieser untersucht werden. Dazu wird im folgenden sowohl die Konstruktion der Fallstudien selbst, als auch die Übertragung der Ergebnisse der Fallstudien auf eine Real-Life Anwendung von Process Mining-Verfahren kritisch betrachtet.

Die in dieser Arbeit durchgeführten Fallstudien basieren auf einer Auswahl von (1) Sicherheitsanforderungen, (2) Process Mining-Verfahren und (3) zwei

Prozessen. Durch die getroffene Auswahl ergibt sich automatisch eine eingeschränkte Betrachtung, so dass weitere Untersuchungen auf Grundlage anderer Gegenstände auch neue und erweiterte Erkenntnisse hervorbringen können. So können beispielsweise auch andere Anforderungen als die im Anforderungskatalog enthaltenen typischen Sicherheitsanforderungen berücksichtigt werden, z. B. Anforderungen sie wie als Bestandteil von Compliance-Regularien auftreten. Auch gibt es im Kontext des Process Minings weitere Verfahren, die Potenzial im Zusammenhang mit der Erkennung von Abweichungen der Prozessausführung versprechen, z. B. Clustering [39, 72] oder Trace Alignment [25]. Die beiden in den Fallstudien betrachteten Prozesse repräsentieren strukturierte und wohldefinierte Prozesse, die Untersuchung weiterer Fallbeispiele für Prozesse mit anderen Charakteristika, z. B. flexible Prozesse, ist ebenfalls vorstellbar.

Sollen die in dieser Arbeit beschriebenen Process Mining-Verfahren in einem Real-Life-Kontext Anwendung finden, so sind zwei wesentliche Hindernisse zu erwarten. Zum einen kann bereits die Erstellung eines für die Überprüfung geeigneten Ereignisprotokolls mit einem angemessenen Reifegrad eine Herausforderung darstellen. Soll das Ereignisprotokoll als Basis für die Überprüfung von Sicherheitsanforderungen dienen, so müssen die Ereignisse des Ereignisprotokolls vollständig und belastbar sein. Somit kommen nur Ereignisprotokolle mit den Reifegraden ★★★★ und ★★★★★ in Frage (cf. Tabelle 3.2). Gegebenenfalls sind nötige Informationen auf mehrere Datenquellen verteilt und müssen vor der Auswertung zunächst extrahiert, zusammengestellt und in ein geeignetes Format transformiert werden. Dazu können Tools wie ProMimport [42], XESame [76] oder Nitro [3] zum Einsatz kommen. In diesem Zusammenhang ist auch zu beachten, dass Ereignisprotokolle in der Regel einen zeitlich begrenzten Ausschnitt der Prozessausführung repräsentieren. So können Traces enthalten sein, deren Anfang, Ende oder beides abgeschnitten ist. Solche Traces müssen (bei fehlendem Ende frühestens zu dem Zeitpunkt zu dem sie vollständig sind) gezielt überprüft werden, damit sie nicht unberücksichtigt bleiben.

Das andere Hindernis stellt die Anwendung der Process Mining-Verfahren mit den bestehenden Software-Applikationen dar. Obwohl die Zahl kommerzieller Applikationen mit integrierten Process Mining Techniken im Bereich des BPM und der Business Process Intelligence (BPI) zunimmt, bleibt die Funktionalität des im Rahmen dieser Arbeit eingesetzten Tools ProM unerreicht [3]. Die Anwendung von ProM (sowohl Version 5.2 als auch 6.1) jedoch erfordert nicht nur eine Einarbeitungszeit, sondern auch Fachwissen, sowohl was die Bedienung als auch die Interpretation der Ergebnisse angeht. Ein Großteil der zur Verfügung stehenden Plugins besitzt etliche Parameter und ist unzureichend dokumentiert. Auch technische Schwierigkeiten können die Arbeit mit ProM behindern, allem voran bei der Verwendung der zahlreichen Plugins, deren prototypische Imple-

mentierung nicht den Anforderungen an einen Gebrauch durch Endanwender gerecht wird. Ein großer Schritt in Richtung Benutzerfreundlichkeit gelingt mit den Process Mining-Tools Nitro und Disco der Firma Fluxicon[1], allerdings liefern diese nicht den nötigen Funktionsumfang zur Überprüfung von Sicherheitsanforderungen wie in dieser Arbeit vorgestellt. Soll also die Anwendung der Verfahren, wie in den Fallstudien demonstriert, in einen industriellen Kontext übertragen werden - so müssen zuvor all diese Hindernisse bewältigt werden.

6.3. Einordnung verwandter Arbeiten

Der Beitrag bestehender Verfahren des Process Minings zu einer *a posteriori* Überprüfung von Sicherheitsanforderungen wurde bisher nur in wenigen Arbeiten thematisiert. Einen groben Überblick über mögliche Richtungen der Unterstützung von Audits durch Process Mining-Verfahren ermöglicht das von van der Aalst et al. [5] entworfene *Auditing Framework* für das sogenannte *Auditing 2.0*. Neben der direkten Betrachtung der Ereignisprotokolle durch Verfahren der Konformitätsprüfung wird unter anderem auch die Betrachtung von extrahierten *de facto* Modellen zur Überprüfung auf Anomalien vorgeschlagen. In dieser Arbeit wurden diese beiden Vorschläge praktisch umgesetzt und gezeigt, dass bei der Verwendung von *de facto* Modellen als Grundlage für die Untersuchung Vorsicht angebracht ist, da bei den konventionellen Verfahren zur Modell-Extraktion mit Informationsverlust zu rechnen ist, so dass Abweichungen unerkannt bleiben können (cf. Abschnitt 5.1.4).

Über die praktische Anwendung von Process Mining-Verfahren für die Überprüfung von Sicherheitsanforderungen in Fallstudien wurde bisher von Jans et al. [52, 53] sowie Accorsi u. Stocker [12] berichtet. Während Jans et al. dabei hervorheben, welche Vorteile ein Audit basierend auf Process Mining-Verfahren gegenüber traditionellen Ansätzen für manuelle Audits besitzt (cf. Kapitel 1), konzentrieren sich Accorsi u. Stocker auf die Überprüfung verschiedener Klassen von Sicherheitsanforderungen mit Verfahren der Konformitätsprüfung und zeigen dabei deren Grenzen bei der Überprüfung von Hypereigenschaften und Potenzial für zukünftige Entwicklungen auf. Die vorliegende Arbeit greift in dem Zusammenhang vor allem den Ansatz von Accorsi u. Stocker auf, wobei durch die hier beschriebenen Fallstudien nicht nur (i) deren Erkenntnisse bestätigt und erweitert werden, sondern auch (ii) Lösungsansätze für die Überprüfung einiger Hypereigenschaften durch die Zusammenführung aller Instanzen im Ereignisprotokoll praktisch demonstriert werden (cf. Abschnitt 5.2.3). Darüber hinaus berücksichtigt diese Arbeit auch (iii) die Verwendung von Process

[1] `http://www.fluxicon.com`

Mining-Verfahren zur Modell-Extraktion für die Kontrollfluss- und Organisationsperspektive (cf. Abschnitt 5.1.4 und 5.2.3) und (iv) die praktische Anwendung des Ansatzes von Baumgrass et al. [21] zu einer erleichterten Überprüfung von RBAC-Policies (cf. Abschnitt 5.2.3).

7. Zusammenfassung und Ausblick

Auf Grundlage der in dieser Arbeit durchgeführten Fallstudien wurde die Eignung von Verfahren des Process Minings für die Compliance-Prüfung mit einem Schwerpunkt auf Anforderungen zur Sicherheit praktisch untersucht. Diesbezüglich konnte gezeigt werden, dass sich die betrachteten Intra-Instanz-Sicherheitsanforderungen bezogen auf Safety-Eigenschaften mit den Verfahren der Log-basierten Verifikation ausnahmslos überprüfen lassen. Darüber hinaus wurde demonstriert, wie eine praktische Einschränkung der bestehenden Verfahren zur Log-basierten Verifikation umgangen werden kann, um zusätzlich auch Inter-Instanz-Sicherheitsanforderungen bezogen auf Hypereigenschaften zu überprüfen.

Bei der Durchführung der beiden Fallstudien wurden nicht nur Verfahren der Konformitätsprüfung zur direkten Überprüfung von Sicherheitsanforderungen auf Ereignisprotokollen eingesetzt, sondern auch Verfahren der Modell-Extraktion angewendet, um Modelle aus dem Ereignisprotokoll zu gewinnen. Eine indirekte Überprüfung von Sicherheitsanforderungen auf Grundlage dieser Modelle war nur eingeschränkt möglich. Obwohl bei der Betrachtung von Modellen zur Organisationsperspektive bestimmte Abweichungen sichtbar gemacht werden konnten, ist für die extrahierten Petri-Netze zur Kontrollflussperspektive das Gegenteil der Fall: die eingebauten Abweichungen wurden aufgrund ihres seltenen Auftretens als Noise deklariert und durch das Modell nicht repräsentiert, so dass sie infolgedessen unerkannt bleiben.

Um das Process Mining über den Kontext dieser Arbeit hinaus für die Überprüfung von Sicherheitsanforderungen einsetzen zu können, ergibt sich Handlungsbedarf in zweierlei Richtungen: der Forschung und der Anwendungsentwicklung. Zum einen ist eine Orientierung auf Datenobjekte im Prozess und ihren Fluss grundsätzlich unabdingbar, da Sicherheitsanforderungen zum Datenaspekt und Datenschutz eine große Rolle spielen. Eine solche Datenperspektive wurde im Zusammenhang des Process Mining bisher nicht explizit berücksichtigt, der Fokus liegt vor allem auf Verfahren zur Kontrollfluss- und Organisationsperspektive. Die Entwicklung von Verfahren zur Extraktion von Informationsflussmodellen kann einen ersten Schritt in diese Richtung ermöglichen. Auch die in der Diskussion vorgeschlagene Erweiterung von Verfahren der Log-basierten Verifikation erfordert weitere Aufmerksamkeit. Zu prüfen ist beispielsweise, ob sich die Klasse

von Hypereigenschaften vollumfänglich durch eine solche Erweiterung erfassen lässt, oder ob Lücken offen bleiben.

Auf dem Gebiet der Anwendungsentwicklung im Bereich des Process Mining müssen zum anderen noch große Schritte in Richtung Benutzerfreundlichkeit gemacht werden, damit sowohl die Bedienung als auch die Interpretation von Ergebnissen auch durch Anwender mit weniger Fachwissen erfolgen kann. Vor einem selbstverständlichen und industriellen Einsatz von Process Mining-Verfahren zur Compliance-Prüfung liegt noch ein weiter Weg.

Literaturverzeichnis

[1] van der Aalst, W. M. P.: Process-aware information systems: Lessons to be learned from process mining. In: *Transactions on Petri Nets and Other Models of Concurrency* 2 (2009), S. 1–26

[2] van der Aalst, W. M. P.: Business Process Simulation Revisited. In: Barjis, J. (Hrsg.) ; van der Aalst, W. M. P. (Hrsg.) ; Mylopoulos, J. (Hrsg.) ; Rosemann, M. (Hrsg.) ; Shaw, M. J. (Hrsg.) ; Szyperski, C. (Hrsg.): *Enterprise and Organizational Modeling and Simulation.* Springer Berlin Heidelberg, 2010, S. 1–14

[3] van der Aalst, W. M. P.: *Process Mining - Discovery, Conformance and Enhancement of Business Processes.* 1. Springer Verlag Berlin Heidelberg, 2011

[4] van der Aalst, W. M. P. ; de Beer, H. T. ; van Dongen, B. F.: Process Mining and Verification of Properties: An Approach based on Temporal Logic. In: *On the Move to Meaningful Internet Systems*, 2005, S. 130–147

[5] van der Aalst, W. M. P. ; van Hee, K. ; van der Werf, J. M. ; Verdonk, M.: Auditing 2.0: Using Process Mining to Support Tomorrow's Auditor. In: *IEEE Computer* 43 (2010), S. 90–93

[6] van der Aalst, W. M. P. ; ter Hofstede, A. H. M. ; Weske, M.: Business Process Management: A Survey. In: *International Conference on Business Process Management*, 2003, S. 1–12

[7] van der Aalst, W. M. P. ; Pesic, M. ; Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. In: *Computer Science - R&D* 23 (2009), S. 99–113

[8] van der Aalst, W. M. P. ; Stahl, C.: *Modeling Business Processes: a Petri Net-Oriented Approach.* 1. The MIT Press, 2011

[9] van der Aalst, W. M. P. ; Weijters, T. ; Maruster, L.: Workflow mining: discovering process models from event logs. In: *IEEE Transactions on Knowledge and Data Engineering* 16 (2004), Nr. 9, S. 1128–1142

[10] Accorsi, R.: Automated Privacy Audits to Complement the Notion of Control for Identity Management. In: de Leeuw, E. (Hrsg.) ; Fischer-Hubner, S. (Hrsg.) ; Tseng, J. (Hrsg.) ; Borking, J. (Hrsg.): *Policies and Research in Identity Management*. Springer-Verlag, 2008

[11] Accorsi, R. ; Sato, Y. ; Kai, S.: Compliance-Monitor zur Frühwarnung vor Risiken. In: *Wirtschaftsinformatik* 50 (2008), Nr. 5, S. 375–382

[12] Accorsi, R. ; Stocker, T.: On the Exploitation of Process Mining for Security Audits: The Conformance Checking Case. In: *ACM Symposium on Applied Computing* (2012)

[13] Accorsi, R. ; Wonnemann, C.: InDico: Information Flow Analysis of Business Processes for Confidentiality Requirements. In: *Security and Trust Management*, 2010, S. 194–209

[14] Accorsi, R. ; Wonnemann, C. ; Dochow, S.: SWAT: A Security Workflow Analysis Toolkit for Reliably Secure Process-aware Information Systems. In: *International Conference on Availability, Reliability and Security*, 2011, S. 692–697

[15] Accorsi, R. ; Wonnemann, C. ; Stocker, T.: Towards Forensic Data Flow Analysis of Business Process Logs. In: *IT Security Incident Management and IT Forensics*, 2011, S. 3–20

[16] Agrawal, R. ; Gunopulos, D. ; Leymann, F.: Mining Process Models from Workflow Logs. In: *Advances in Database Technology*, 1998, S. 469–483

[17] Alpern, B. ; Schneider, F. B.: Recognizing safety and liveness. In: *Distributed Computing* 2 (1987), S. 117–126

[18] Arsac, W. ; Compagna, L. ; Pellegrino, G. ; Ponta, S.: Security Validation of Business Processes via Model-Checking. In: Erlingsson, U. (Hrsg.) ; Wieringa, R. (Hrsg.) ; Zannone, N. (Hrsg.): *Engineering Secure Software and Systems*. Springer Berlin / Heidelberg, 2011, S. 29–42

[19] Atluri, V. ; Warner, J.: Security for Workflow Systems. In: Gertz, M. (Hrsg.) ; Jajodia, S. (Hrsg.) ; Rutgers University Newark NJ (Veranst.): *Handbook of Database Security*. Springer US, 2008, S. 213–230

[20] Baier, Christel ; Katoen, Joost-Pieter: *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008

[21] Baumgrass, A. ; Baier, T. ; Mendling, J. ; Strembeck, M.: Conformance Checking of RBAC Policies in Process-Aware Information Systems. In: *International Conference on Business Process Management Workshops*, 2011

[22] de Beer, H. T. ; van den Brand, P. C. W.: *The LTL Checker Plugins.* `https://svn.win.tue.nl/repos/prom/Documentation/LTL-Manual/LTLChecker-Manual.ps`

[23] Bezerra, F. ; Wainer, J.: Anomaly detection algorithms in logs of process aware systems. In: *ACM Symposium on Applied Computing*, 2008, S. 951–952

[24] Bezerra, F. ; Wainer, J. ; van der Aalst, W. M. P.: Anomaly Detection using Process Mining. In: *Enterprise, Business-Process and Information Systems Modeling*, 2009, S. 149–161

[25] Bose, R. P. J. C. ; van der Aalst, W. M. P.: Process Diagnostics Using Trace Alignment: Opportunities, Issues, and Challenges. In: *Information Systems* 37 (2012), S. 117–141

[26] Burattin, A. ; Sperduti, A.: PLG: A Framework for the Generation of Business Process Models and Their Execution Logs. In: *International Conference on Business Process Management Workshops*, 2011, S. 214–219

[27] Carlin, A. ; Gallegos, F.: IT Audit: A Critical Business Process. In: *IEEE Computer* 40 (2008), Nr. 7, S. 87–89

[28] Charfi, A. ; Mezini, M.: Aspect-oriented workflow languages. In: *On the Move to Meaningful Internet Systems*, 2006, S. 183–200

[29] Clarkson, M. R. ; Schneider, F. B.: Hyperproperties. In: *Journal of Computer Security*, IOS Press, 2010, S. 1157–1210

[30] Cook, J. E. ; Wolf, A. L.: Discovering Models of Software Processes from Event-Based Data. In: *ACM Trans. Softw. Eng. Methodol.* 7 (1998), Nr. 3, S. 215–249

[31] Datta, A.: Automating the Discovery of AS-IS Business Process Models: Probabilistic and Algorithmic Approaches. In: *Information Systems Research* 9 (1998), Nr. 3, S. 275–301

[32] Denning, D. E. ; Denning, P. J.: Certification of Programs for Secure Information Flow. In: *Communications of the ACM* 20 (1977), S. 1–10

[33] van Dongen, B. F.: *Process Mining and Verification*, TU Eindhoven, Diss., 2006

[34] van Dongen, B. F. ; van der Aalst, W. M. P.: A Meta Model for Process Mining Data. In: *Enterprise Modelling and Ontologies for Interoperability*, 2005

[35] van Dongen, B. F. ; Alves de Medeiros, A. K. ; Verbeek, H. M. W. ; Weijters, A. J. M. M. ; van der Aalst, W. M. P.: The ProM framework: A new era in process mining tool support. In: *International Conference on Applications and Theory of Petri Nets*, 2005, S. 444–454

[36] van Dongen, B. ; Alves de Medeiros, A. K. ; Wen, L.: Process Mining: Overview and Outlook of Petri Net Discovery Algorithms. In: Jensen, K. (Hrsg.) ; van der Aalst, W. M. P. (Hrsg.): *Transactions on Petri Nets and Other Models of Concurrency II*. Springer Verlag Berlin/Heidelberg, 2009, S. 225–242

[37] Dumas, M. ; van der Aalst, W. M. P. ; ter Hofstede, A. H.: *Process Aware Information Systems: Bridging People and Software Through Process Technology*. 1. Wiley-Interscience, 2005

[38] Ferraiolo, D. F. ; Kuhn, D. R.: Role-Based Access Controls. In: *National Computer Security Conference*, 1992, S. 554–563

[39] Ghionna, L. ; Greco, G. ; Guzzo, A. ; Pontieri, L.: Outlier detection techniques for process mining applications. In: *International Symposium on Foundations of Intelligent Systems*, 2008, S. 150–159

[40] Golani, M. ; Pinter, S. S.: Generating a Process Model from a Process Audit Log. In: *International Conference on Business Process Management*, 2003, S. 136–151

[41] Greco, G. ; Guzzo, A. ; Pontieri, L. ; Saccà, D.: Discovering Expressive Process Models by Clustering Log Traces. In: *IEEE Trans. Knowl. Data Eng.* 18 (2006), Nr. 8, S. 1010–1027

[42] Günther, C. ; van der Aalst, W. M. P.: A Generic Import Framework for Process Event Logs. In: Eder, J. (Hrsg.) ; Dustdar, S. (Hrsg.): *Business Process Management Workshops*. Springer Berlin / Heidelberg, 2006, S. 81–92

[43] Günther, C. W.: *OpenXES*. `http://www.xes-standard.org/_media/openxes/developerguide7.pdf`

[44] Günther, C. W.: *Process Mining in Flexible Environments*, TU Eindhoven, Diss., 2009

[45] Hansen, H. R. ; Neumann, G. ; Bea, F. X. (Hrsg.) ; Schweitzer, M. (Hrsg.): *Wirtschaftsinformatik 1.* 10. Lucius & Lucius - Stuttgart, 2005

[46] Herbst, Joachim: A Machine Learning Approach to Workflow Management. In: *European Conference on Machine Learning*, 2000, S. 183–194

[47] IEEE Task Force on Process Mining: *Process Mining Manifesto.* `http://www.win.tue.nl/ieeetfpm/doku.php?id=shared:process_mining_manifesto`

[48] ISO - International Organization for Standardization: *Systems and software engineering – High-level Petri nets – Part 2: Transfer format.* `http://www.iso.org/iso/catalogue_detail.htm?csnumber=43538`

[49] ITU-T: *Security Frameworks for Open Systems: Overview.* 1995

[50] Jablonski, S. ; Bussler, C.: Workflow management: modeling concepts, architecture and implementation. In: *International Thomson Computer Press* (1996)

[51] Jans, M. ; Alles, M.: Process Mining in Auditing: From Current Limitations to Future Challenges. In: *International Conference on Business Process Management*, 2011, S. 394–397

[52] Jans, M. ; Alles, M. ; Vasarhelyi, M.: Process Mining of Event Logs in Internal Auditing: A case study. In: *International Symposium on Accounting Information Systems*, 2011

[53] Jans, M. ; Depaire, B. ; Vanhoof, K.: Does Process Mining Add to Internal Auditing? An Experience Report. In: *Enterprise, Business-Process and Information Systems Modeling*, 2011, S. 31–45

[54] Lamport, L.: Proving the Correctness of Multiprocess Programs. In: *IEEE Trans. Softw. Eng.* 3 (1977), Nr. 2, S. 125–143

[55] Lowis, L.: *Automatisierte Compliance-Prüfung von Geschäftsprozessen*, Albert-Ludwigs-Universität Freiburg im Breisgau, Diss., 2011

[56] Alves de Medeiros, A. K.: *Genetic Process Mining*, TU Eindhoven, Diss., 2006

[57] Alves de Medeiros, A. K. ; Günther, C. W.: Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms. In: *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, 2005, S. 177–190

[58] Montali, M.: *Lecture Notes in Business Information Processing*. Bd. 56: *Specification and Verification of Declarative Open Interaction Models - A Logic-Based Approach*. Springer, 2010

[59] Müller, G. ; Accorsi, R. ; Höhn, S. ; Sackmann, S.: Sichere Nutzungskontrolle für mehr Transparenz in Finanzmärkten. In: *Informatik-Spektrum* 33 (2010), S. 3–13

[60] Müller, Günter ; Terzidis, O.: IT-Compliance und IT-Governance. In: *Wirtschaftsinformatik* 50 (2008), Nr. 5, S. 341–343

[61] Müller, J. ; Mülle, J. ; von Stackelberg, S. ; Bohm, K.: Secure Business Processes in Service-Oriented Architectures – a Requirements Analysis. In: *IEEE European Conference on Web Services*, IEEE, 2010, S. 35–42

[62] Park, Jaehong ; Sandhu, Ravi S.: Towards usage control models: beyond traditional access control. In: *ACM Symposium on Access Control Models and Technologies*, 2002, S. 57–64

[63] Pérez-Castillo, R. ; Sanchez-Gonzalez, L. ; Piattini, M. ; Garcia, F. ; Garcia-Rodriguez De Guzman, I.: *Obtaining Thresholds for the Effectiveness of Business Process Mining*. 2011

[64] Rozinat, A.: *Process Mining: Conformance and Extension*, TU Eindhoven, Diss., 2010

[65] Rozinat, A. ; van der Aalst, W. M. P.: Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. In: *International Conference on Business Process Management*, 2005, S. 163–176

[66] Rozinat, A. ; van der Aalst, W. M. P.: Decision Mining in ProM. In: *International Conference on Business Process Management*, 2006, S. 420–425

[67] Rozinat, A. ; Wynn, M. ; van der Aalst, W. M. P. ; Hofstede, A. ; Fidge, C.: Workflow Simulation for Operational Decision Support Using

Design, Historic and State Information. In: *International Conference on Business Process Management*. Berlin, Heidelberg : Springer-Verlag, 2008, S. 196–211

[68] Runeson, P. ; Höst, M.: Guidelines for conducting and reporting case study research in software engineering. In: *Empirical Software Engineering* 14 (2008), Nr. 2, S. 131–164

[69] Schimm, Guido: Mining Most Specific Workflow Models from Event-Based Data. In: *International Conference on Business Process Management*, 2003, S. 25–40

[70] Schulte-Zurhausen, M.: *Organisation*. 4. Verlag Franz Vahlen München, 2005

[71] Song, M. ; van der Aalst, W. M. P.: Towards Comprehensive Support for Organizational Mining. In: *Decision Support Systems* 46 (2008), S. 300–317

[72] Song, M. ; Günther, C. W. ; van der Aalst, W. M. P.: Trace clustering in process mining. In: *International Conference on Business Process Management*, 2008, S. 109–120

[73] von Stackelberg, S.: *Sicherheit in Workflow-Management-Systemen*. `http://dbis.ipd.uni-karlsruhe.de/download/Secure-Workflows.pdf`

[74] Stocker, T.: Data Flow-oriented Process Mining to Support Security Audits. In: *PhD Symposium of the International Conference of Service Oriented Computing*, 2011

[75] Verbeek, H. M. W. ; Buijs, J. C. A. M. ; van Dongen, B. F. ; van der Aalst, W. M. P.: ProM 6: The Process Mining Toolkit. In: *International Conference on Business Process Management*, 2010

[76] Verbeek, H. M. W. ; Buijs, J. C. A. M. ; van Dongen, B. ; van der Aalst, W. M. P.: XES, XESame and ProM6. In: *Information Systems Evolution (CAiSE Forum)* (2010), S. 60–75

[77] Weijters, A. J. M. M. ; van der Aalst, W. M. P.: Rediscovering Workflow Models from Event-Based Data using Little Thumb. In: *Integrated Computer-Aided Engineering* 10 (2001)

[78] Workflow Management Coalition: *Workflow Security Considerations.* http://www.wfmc.org/Download-document/WFMC-TC-1019-Ver-1-Workflow-Security-Considerations.html

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

B2B	Business-to-Business
BPEL	Business Process Execution Language
BPI	Business Process Intelligence
BPM	Business Process Management
BPMN	Business Process Modeling Notation
CRM	Customer Relationship Management
ERP	Enterprise Resource Planning
GLB	Gramm-Leach-Bliley Act
HIPAA	Health Information Portability and Accountability Act
IT	Informationstechnologie
MXML	Mining eXtensible Markup Language
PAIS	Process-aware information system
SOX	Sarbanes-Oxley Act
WfMS	Workflow-Management System
XES	eXtensible Event Stream
YAWL	Yet Another Workflow Language

Anhang

A. LTL Sprachbestandteile

Tabelle A.1.: Übersicht über die Bestandteile der Sprache für die log-basierte Verifikation. Quelle: Handbuch *The LTL Checker plugins* [22].

Typen für Variablen	
`number`	Float Zahl
`string`	Zeichenkette
`set`	Menge
`date`	Datum
Vergleichsoperatoren	
`==`	Äquivalenz
`!=`	Nichtäquivalenz
`<=`	kleiner oder gleich
`>=`	größer oder gleich
`<`	kleiner
`>`	größer
Aussagenlogik	
`!(A)`	Nicht
`(A /\B)`	Logisches Und
`(A \/ B)`	Logisches Oder
`(A -> B)`	Implikation
`(A <-> B)`	Biimplikation
Quantifizierung	
`forall [ i: SetAttribute | Ai ]`	Universelle Quantifizierung
`exists [ i: SetAttribute | Ai ]`	Existenzielle Quantifizierung
Lineare temporale Logik	
`_O(A)`	nexttime
`<>(A)`	eventually
`[](A)`	always
`A _U B`	until
Erweitert	
`[ @http://ontology#Concept ]`	Semantische Erweiterung auf Ontologien

B. Generierung von Ereignisprotokollen

B.1. Einsetzen von Instanzattributen

Listing B.1: Perl-Skript zum Einsetzen von Instanzattributen.

```
1  #!/usr/bin/perl
2  #
3  # Author: Meike Ullrich
4  # Date: Wed 15 Mar 2012 15:43:11 CET
5
6  use strict;
7  use XML::DOM;
8  my $parser = new XML::DOM::Parser;
9
10 # Einlesen des Ereignisprotokolls
11 my $doc = $parser->parsefile ("input.MXML");
12
13 my $PIs = $doc->getElementsByTagName('ProcessInstance');
14 my $n = $PIs->getLength();
15
16 # Betrachten der einzelnen Prozessinstanzen
17 for (my $i = 0; $i < $n; $i++) {
18    my $PI = $PIs->item($i);
19
20    # Auslesen der Aktivitaet des zweiten AuditTrailEntries (ATE)
21    my $ATEs = $PI->getElementsByTagName("AuditTrailEntry");
22    my $ATE_2 = $ATEs->item(1);
23    my $WMEs = $ATE_2->getElementsByTagName("WorkflowModelElement");
24    my $WME = $WMEs->item(0);
25    my $activity = $WME->getFirstChild()->getData();
26
27    # Wird Aktivitaet Register as Low-value claim (B) ausgefuehrt,
28    # wird ein zufaelliger Wert von 1-4999 fuer den Betrag erzeugt
29    my $value;
30    if ($activity eq 'B') {
31       $value = int(rand(4999)) + 1;
32    }
33    # Wird Aktivitaet Register as High-value claim (C) ausgefuehrt,
34    # wird ein Wert von 5000-500000 fuer den Betrag erzeugt
35    elsif ($activity eq 'C') {
36       $value = int(rand(495000)) + 5000;
37    }
```

```
38      # Andernfalls wird eine Warnung ausgegeben
39      else {
40          print "Warning, Path not recognized!\n";
41      }
42
43      # Einbinden des Data-Elements in der Prozessinstanz sowie des
44      # Erstellung des Attributs Betrag mit dem oben generierten Wert
45      my $Data = $PI->appendChild($doc->createElement("Data"));
46      my $Att = $Data->appendChild($doc->createElement("Attribute"));
47      $Att->appendChild($doc->createTextNode($value));
48      $Att->setAttribute('name','betrag');
49  }
50
51  # Ausgabe des Ereignisprotokolls
52  $doc->printToFile ("output.MXML");
53  $doc->dispose;
```

B.2. Simulationsparameter Prozess Schadenersatzanspruch

Tabelle B.1.: Simulationsparameter für den Prozess Schadenersatzanspruch.

Generelle Angaben
◦ Anzahl der zu generierenden Traces: *500* ◦ Datum des Beginns des Logs: *1. Januar 2012* ◦ Anzahl von vervollständigten Fällen pro Tag: *10*
Kontrollfluss
◦ Siehe Petri-Netz in Abb. 5.1 ◦ Wahrscheinlichkeiten für Verzweigung an XOR-Split: $A \rightarrow B : 0.5, A \rightarrow C : 0.5$
Zeit
◦ Dauer Aktivität A, B, C, D, E, F, G, H, I: *10 Minuten* (Abweichung: *0.01*)
Ressource/Organisation
◦ Liste von Benutzern: *{1}* ◦ Liste von Rollen: *{generic}* ◦ Zuordnung von Rollen zu Benutzern: *generic: {1}* ◦ Zuordnung von Rollen zu Aktivitäten: *generic: {A, B, C, D, E, F, G,H,I}*
Daten
◦ Instanzattribute werden durch das Perl-Skript in Listing B.1 eingesetzt.

B.3. Simulationsparameter Prozess Kreditvergabe

Tabelle B.2.: Simulationsparameter für den Prozess Kreditvergabe.

Generelle Angaben
◦ Anzahl der zu generierenden Traces: *200* ◦ Datum des Beginns des Logs: *1. Januar 2012* ◦ Anzahl von vervollständigten Fällen pro Tag: *10* (Abweichung: *0.1*)
Kontrollfluss
◦ Siehe Petri-Netz in Abb. 5.6
Zeit
◦ Dauer Aktivität A: *10 Minuten* (Abweichung: *0.1*) ◦ Dauer Aktivität B: *10 Minuten* (Abweichung: *0.1*) ◦ Dauer Aktivität C: *10 Minuten* (Abweichung: *0.1*) ◦ Dauer Aktivität D: *10 Minuten* (Abweichung: *0.1*) ◦ Dauer Aktivität E: *10 Minuten* (Abweichung: *0.1*) ◦ Dauer Aktivität F: *10 Minuten* (Abweichung: *0.1*) ◦ Dauer Aktivität G: *10 Minuten* (Abweichung: *0.1*)
Ressource/Organisation
◦ Liste von Benutzern: $\{A_1, \ldots, A_{10}, B_1, \ldots, B_{10}\}$ ◦ Liste von Rollen: *{clerk_preprocessor, clerk_postprocessor, bank_manager}* ◦ Zuordnung von Rollen zu Benutzern: *clerk_preprocessor:* $\{A_1,A_2,A_3,A_4,B_1,B_2,B_3,B_4\}$, *clerk_postprocessor:* $\{A_5,A_6,A_7,A_8,B_5,B_6,B_7,B_8\}$, *bank_manager:* $\{A_9,A_{10},B_9,B_{10}\}$ ◦ Zuordnung von Rollen zu Aktivitäten: *clerk_preprocessor: {A, B}, clerk_postprocessor: {C, D, E, G}, bank_manager: {F}*
Daten
◦ Ereignisattribute: *Kunde, Abteilung, Eingabe, Ausgabe, Stufe*

C. Listings

C.1. LTL-Formeln für das Fallbeispiel Schadenersatzanspruch

Listing C.1: Schadenersatzanspruch.ltl: LTL-Formeln zur Überprüfung der Sicherheitsanforderungen **(A1a)**, **(A1b)**, **(A2a)**, **(A3)** und **(A4)**.

```
1  # Defining attributes
2  set ate.WorkflowModelElement;
3  number pi.betrag;
4
5  # Renaming Attributes
6  rename ate.WorkflowModelElement as activity;
7  rename pi.betrag as betrag;
8
9  # Bedingung zur Sicherheitsanforderung (A1a)
10 formula bedingung_a1a () :=
11 {
12    <h2>Aktivitaet Reject Claim (X) darf nicht enthalten sein</h2>
13 }
14 !(<>( activity == "X" ));
15
16
17 # Bedingung zur Sicherheitsanforderung (A1b)
18 formula bedingung_a1b () :=
19 {
20    <h2>Aktivitaet Check Policy (D) muss enthalten sein</h2>
21 }
22 <>( activity == "D" );
23
24 # Bedingung zur Sicherheitsanforderung (A2a)
25 formula bedingung_a2a () :=
26 {
27    <h2>Falls Complete High-value Claim (F) ausgefuehrt wird, muss
28    zuvor Check Liability (H) ausgefuehrt worden sein</h2>
29 }
30 ( <>( ( activity == "H" /\ <>( activity == "F" ) ) ) \/
31                !(<> ( activity == "F" ) ) );
32
33 # last() is a formula to point to the last state. The last state
34 # is that state in which in the next state a comparison of two
35 # equal attributes is not true.
36 subformula last( ) :=
37 {}
```

```
38 !( _O (  activity == activity ) );
39
40 # Bedingung zur Sicherheitsanforderung (A3)
41 formula bedingung_a3 () :=
42 {
43 }
44 (((
45    # A B D E I oder
46    (activity == "A" /\ _O((activity == "B" /\ _O((activity == "D" /\
47    _O((activity == "E" /\ _O((activity == "I" /\ last()))))))))))
48 \/
49    # A C D G H F I oder
50    (activity == "A" /\ _O((activity == "C" /\ _O((activity == "D" /\
51    _O((activity == "G" /\ _O((activity == "H" /\ _O((activity =="F" /\
52    _O((activity == "I" /\ last ())))))))))))))
53 )
54 \/
55    # A C G D H F I oder
56    (activity == "A" /\ _O((activity == "C" /\ _O((activity == "G" /\
57    _O((activity == "D" /\ _O((activity == "H" /\ _O((activity =="F" /\
58    _O((activity == "I" /\ last ())))))))))))))
59 )
60 \/
61    # A C G H D F I
62    (activity == "A" /\ _O((activity == "C" /\ _O((activity == "G"  /\
63    _O((activity == "H" /\ _O((activity == "D" /\ _O((activity =="F" /\
64    _O((activity == "I" /\ last ())))))))))))))
65 );
66
67 # Bedingung zur Sicherheitsanforderung (A4)
68 formula bedingung_a4 () :=
69 {
70    <h2>Auf Set Checkpoint Start (A) folgt direkt Register as High-
71    value Claim (C) falls Instanzattribut Betrag >= 5000,
72    anderenfalls Register as Low-value Claim (B)</h2>
73
74 }
75 ((
76    (<>((activity == "A" /\ _O(activity == "C") ) ) /\ betrag >= 5000)
77 \/
78    (<>((activity == "A" /\ _O(activity == "B") ) ) /\ betrag < 5000)
79 )
80 \/
81    !(<>( activity == "A" ))
82 );
```

C.2. LTL-Formeln für das Fallbeispiel Kreditvergabe

Listing C.2: Kreditvergabe_01.ltl: LTL-Formeln zur Überprüfung der Sicherheitsanforderungen **(A6a)**, **(A7a)**, **(A8)** und **(A9)**.

```
# Defining attributes
set ate.WorkflowModelElement;
set ate.Originator;
set ate.Abteilung;
string ate.Stufe;

# Renaming Attributes
rename ate.WorkflowModelElement as activity;
rename ate.Originator as teilnehmer;
rename ate.Abteilung as abteilung;
rename ate.Stufe as stufe;

# Unterformel fuer Teilnehmer T fuehrt Aktivitaet A aus
subformula T_does_A( T: teilnehmer , A: activity ) := {}
( teilnehmer == T /\ activity == A);

subformula eventually_T_does_A ( T: teilnehmer, A: activity ) := {}
<>( T_does_A( T, A ) );

# Bedingung zur Sicherheitsanforderung (A6a)
formula bedingung_a6a () :=
{
    <h2>Sign Form (F) darf nur von Teilnehmern
    A_9, A_10, B_9, B_10 ausgefuehrt werden.</h2>
}
(
    ((( <>( T_does_A( "A_9", "F" ) ) \/ <>( T_does_A( "A_10", "F" )))
    \/ <>( T_does_A( "B_9", "F" )) ) \/ <>( T_does_A( "B_10", "F" )))
    \/ !( <> ( activity == "F" ) )
);

# Bedingung zur Sicherheitsanforderung (A7a)
formula bedingung_a7a () :=
{
    <h2>Wird Select Bundle Product (E) von Teilnehmer t ausgefuehrt,
    darf Sign Form (F) nicht von Teilnehmer t ausgefuehrt werden.</h2>
}
!(exists[ t: teilnehmer |
    ( eventually_T_does_A( t, "E" ) /\
         eventually_T_does_A( t, "F" ) ) ]);

# Bedingung zur Sicherheitsanforderung (A8)
formula bedingung_a8 () :=
{
    <h2>Customer Identification (B) darf nur von Teilnehmer 1
    mit Datenzugriff auf Daten "secret" ausgefuehrt werden</h2>
}
!(<> (( activity == "B" /\ ( stufe == "secret"
                /\ teilnehmer != "A_1" ))));
```

```
50
51 # Bedingung zur Sicherheitsanforderung (A9)
52 formula bedingung_a9 () :=
53 {
54     <h2>An einer Instanz duerfen nur Teilnehmer der Abteilung
55     A oder der Abteilung B teilnehmen</h2>
56 }
57 !(( <>( abteilung == "B" ) /\ <> ( abteilung == "A")));
```

Listing C.3: Kreditvergabe_02.ltl: LTL-Formeln zur Überprüfung der Sicherheitsanforderungen **(A7b)** und **(A10)**. Zu verwenden nur auf einem Ereignisprotokoll in dem alle Instanzen zu einer einzigen Instanz zusammengeführt wurden.

```
1  # Defining attributes
2  set ate.WorkflowModelElement;
3  set ate.Originator;
4  set ate.Kunde;
5  set ate.Eingabe;
6  set ate.Ausgabe;
7  set ate.Abteilung;
8
9  # Renaming Attributes
10 rename ate.WorkflowModelElement as activity;
11 rename ate.Originator as teilnehmer;
12 rename ate.Kunde as kunde;
13 rename ate.Eingabe as eingabe;
14 rename ate.Ausgabe as ausgabe;
15 rename ate.Abteilung as abteilung;
16
17 ########################################################################
18 # Diese Formeln funktionieren nur auf einem Ereignisprotokoll in dem
19 # alle Instanzen zu einer einzigen Instanz zusammengefuehrt wurden.
20 ########################################################################
21
22 # Bedingung zur Sicherheitsanforderung (A7b)
23 formula bedingung_a7b () :=
24 {
25     <h2>Prueft, ob kein Kunde mehrmals durch Aktivitaet A
26     aufgenommen wird.</h2>
27 }
28 !(exists[ k: kunde |
29       <> ( ( (activity == "A" /\ kunde == k)
30          /\ <> ( (kunde == k /\ activity == "A")) ) )
31 ]);
32
33 # Ereignis von Abteilung A liest Objekt E
34 subformula eingabe_object (A: abteilung, E: eingabe):=
35 {}
36 (abteilung == A /\ eingabe == E );
37
38 # Ereignis von Abteilung A schreibt Objekt O
39 subformula ausgabe_object (A: abteilung, O: ausgabe):=
```

```
40 {}
41 ( abteilung == A /\ ausgabe == O);
42
43 # Schreiben von Objekt O und anschliessendes Lesen von Objekt I
44 # so dass I=O und unterschiedliche Abteilungen
45 subformula dataflow_from_A_to_B(A: abteilung, B: abteilung,
46                 E: eingabe, O: ausgabe):=
47 {}
48 <>( ( ausgabe_object(A,O) /\ <> ( ( eingabe_object(B, E) /\
49                     (E==O /\ A!=B )))));
50
51 # Bedingung zur Sicherheitsanforderung (A10)
52 formula bedingung_a10 () :=
53 {
54     <h2>Prueft, ob kein expliziter Informationsfluss zwischen zwei
55     verschiedenen Abteilungen stattfindet</h2>
56 }
57 !(
58 exists [ a: abteilung |
59     exists[ b: abteilung |
60     exists[ o: ausgabe |
61         dataflow_from_A_to_B(a,b,o,o)
62     ]
63     ]
64 ]);
```

C.3. RBAC-Modell in XML für das Fallbeispiel Kreditvergabe

Listing C.4: Kreditvergabe_RBAC.xml: Generiert LTL-Formeln für die Überprüfung der Sicherheitsanforderungen **(A6a)**, **(A6b)** und **(A7a)**.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <RBACModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="./process-related-RBAC-schema.xsd">
3     <processtype id="PT01" name="Kreditvergabe" xsi:type="
          BusinessActivity">
4        <node id="BA01" xsi:type="BusinessAction" name="A">
5           <outgoing resource="AE01"/>
6        </node>
7        <node id="BA02" xsi:type="BusinessAction" name="B">
8           <incoming resource="AE01"/>
9           <outgoing resource="AE02"/>
10       </node>
11       <node id="FN01" xsi:type="ForkNode">
12          <incoming resource="AE02"/>
13          <outgoing resource="AE03"/>
14          <outgoing resource="AE04"/>
15       </node>
16       <node id="BA03" xsi:type="BusinessAction" name="C">
17          <incoming resource="AE03"/>
```

```
18          <outgoing resource="AE05"/>
19       </node>
20       <node id="BA04" xsi:type="BusinessAction" name="D">
21          <incoming resource="AE04"/>
22          <outgoing resource="AE06"/>
23       </node>
24       <node id="JN01" xsi:type="JoinNode">
25          <incoming resource="AE05"/>
26          <incoming resource="AE06"/>
27          <outgoing resource="AE07"/>
28       </node>
29       <node id="BA05" xsi:type="BusinessAction" name="E">
30          <incoming resource="AE07"/>
31          <outgoing resource="AE08"/>
32       </node>
33       <node id="BA06" xsi:type="BusinessAction" name="F">
34          <incoming resource="AE08"/>
35          <outgoing resource="AE09"/>
36          <dynamicExclusion resource="BA05"/>
37       </node>
38       <node id="BA07" xsi:type="BusinessAction" name="G">
39          <incoming resource="AE09"/>
40       </node>
41       <edge id="AE01" source="BA01" target="BA02" xsi:type="
             ControlFlow"/>
42       <edge id="AE02" source="BA02" target="FN01" xsi:type="
             ControlFlow"/>
43       <edge id="AE03" source="FN01" target="BA03" xsi:type="
             ControlFlow"/>
44       <edge id="AE04" source="FN01" target="BA04" xsi:type="
             ControlFlow"/>
45       <edge id="AE05" source="BA03" target="JN01" xsi:type="
             ControlFlow"/>
46       <edge id="AE06" source="BA04" target="JN01" xsi:type="
             ControlFlow"/>
47       <edge id="AE07" source="JN01" target="BA05" xsi:type="
             ControlFlow"/>
48       <edge id="AE08" source="BA05" target="BA06" xsi:type="
             ControlFlow"/>
49       <edge id="AE09" source="BA06" target="BA07" xsi:type="
             ControlFlow"/>
50    </processtype>
51    <role id="R01" name="Clerk_Preprocessor" xsi:type="Role">
52       <ownedTask resource="BA01"/>
53       <ownedTask resource="BA02"/>
54    </role>
55    <role id="R02" name="Clerk_Postprocessor" xsi:type="Role">
56       <ownedTask resource="BA03"/>
57       <ownedTask resource="BA04"/>
58       <ownedTask resource="BA05"/>
59       <ownedTask resource="BA07"/>
60    </role>
61    <role id="R03" name="Bank_manager" xsi:type="Role">
62       <ownedTask resource="BA01"/>
63       <ownedTask resource="BA02"/>
64       <ownedTask resource="BA03"/>
```

```
65          <ownedTask resource="BA04"/>
66          <ownedTask resource="BA05"/>
67          <ownedTask resource="BA06"/>
68          <ownedTask resource="BA07"/>
69       </role>
70       <subject id="S01" name="A_1" xsi:type="Subject">
71          <ownedRole resource="R01"/>
72       </subject>
73       <subject id="S02" name="A_2" xsi:type="Subject">
74          <ownedRole resource="R01"/>
75       </subject>
76       <subject id="S03" name="A_3" xsi:type="Subject">
77          <ownedRole resource="R01"/>
78       </subject>
79       <subject id="S04" name="A_4" xsi:type="Subject">
80          <ownedRole resource="R01"/>
81       </subject>
82       <subject id="S05" name="A_5" xsi:type="Subject">
83          <ownedRole resource="R02"/>
84       </subject>
85       <subject id="S06" name="A_6" xsi:type="Subject">
86          <ownedRole resource="R02"/>
87       </subject>
88       <subject id="S07" name="A_7" xsi:type="Subject">
89          <ownedRole resource="R02"/>
90       </subject>
91       <subject id="S08" name="A_8" xsi:type="Subject">
92          <ownedRole resource="R02"/>
93       </subject>
94       <subject id="S09" name="A_9" xsi:type="Subject">
95          <ownedRole resource="R03"/>
96       </subject>
97       <subject id="S10" name="A_10" xsi:type="Subject">
98          <ownedRole resource="R03"/>
99       </subject>
100      <subject id="S11" name="B_1" xsi:type="Subject">
101         <ownedRole resource="R01"/>
102      </subject>
103      <subject id="S12" name="B_2" xsi:type="Subject">
104         <ownedRole resource="R01"/>
105      </subject>
106      <subject id="S13" name="B_3" xsi:type="Subject">
107         <ownedRole resource="R01"/>
108      </subject>
109      <subject id="S14" name="B_4" xsi:type="Subject">
110         <ownedRole resource="R01"/>
111      </subject>
112      <subject id="S15" name="B_5" xsi:type="Subject">
113         <ownedRole resource="R02"/>
114      </subject>
115      <subject id="S16" name="B_6" xsi:type="Subject">
116         <ownedRole resource="R02"/>
117      </subject>
118      <subject id="S17" name="B_7" xsi:type="Subject">
119         <ownedRole resource="R02"/>
120      </subject>
```

```
121     <subject id="S18" name="B_8" xsi:type="Subject">
122        <ownedRole resource="R02"/>
123     </subject>
124     <subject id="S19" name="B_9" xsi:type="Subject">
125        <ownedRole resource="R03"/>
126     </subject>
127     <subject id="S20" name="B_10" xsi:type="Subject">
128        <ownedRole resource="R03"/>
129     </subject>
130     <relationship id="R2S01" xsi:type="RoleToSubjectAssignment" role="
            R01" subject="S01"/>
131     <relationship id="R2S02" xsi:type="RoleToSubjectAssignment" role="
            R01" subject="S02"/>
132     <relationship id="R2S03" xsi:type="RoleToSubjectAssignment" role="
            R01" subject="S03"/>
133     <relationship id="R2S04" xsi:type="RoleToSubjectAssignment" role="
            R01" subject="S04"/>
134     <relationship id="R2S05" xsi:type="RoleToSubjectAssignment" role="
            R02" subject="S05"/>
135     <relationship id="R2S06" xsi:type="RoleToSubjectAssignment" role="
            R02" subject="S06"/>
136     <relationship id="R2S07" xsi:type="RoleToSubjectAssignment" role="
            R02" subject="S07"/>
137     <relationship id="R2S08" xsi:type="RoleToSubjectAssignment" role="
            R02" subject="S08"/>
138     <relationship id="R2S09" xsi:type="RoleToSubjectAssignment" role="
            R03" subject="S09"/>
139     <relationship id="R2S10" xsi:type="RoleToSubjectAssignment" role="
            R03" subject="S10"/>
140     <relationship id="R2S11" xsi:type="RoleToSubjectAssignment" role="
            R01" subject="S11"/>
141     <relationship id="R2S12" xsi:type="RoleToSubjectAssignment" role="
            R01" subject="S12"/>
142     <relationship id="R2S13" xsi:type="RoleToSubjectAssignment" role="
            R01" subject="S13"/>
143     <relationship id="R2S14" xsi:type="RoleToSubjectAssignment" role="
            R01" subject="S14"/>
144     <relationship id="R2S15" xsi:type="RoleToSubjectAssignment" role="
            R02" subject="S15"/>
145     <relationship id="R2S16" xsi:type="RoleToSubjectAssignment" role="
            R02" subject="S16"/>
146     <relationship id="R2S17" xsi:type="RoleToSubjectAssignment" role="
            R02" subject="S17"/>
147     <relationship id="R2S18" xsi:type="RoleToSubjectAssignment" role="
            R02" subject="S18"/>
148     <relationship id="R2S19" xsi:type="RoleToSubjectAssignment" role="
            R03" subject="S19"/>
149     <relationship id="R2S20" xsi:type="RoleToSubjectAssignment" role="
            R03" subject="S20"/>
150 </RBACModel>
```