cLog: Complex Event Log Synthesis Tool

Julius Holderer

Albert-Ludwigs-Universität Freiburg, Abt. Telematik, 79098 Freiburg, holderer@iig.uni-freiburg.de

Abstract

Geschäftsprozessmanagement (GPM) im Großen weckt hinsichtlich der Geschäftsprozesssicherheit einen Bedarf an Sicherheitsmechanismen, wie Monitore und Audits, die in Zukunft nicht einen oder isolierte Prozesse analysieren - wie es derzeit der Fall ist -, sondern mit vielen vernetzten Prozessen umgehen können müssen. GPM im Großen zeichnet sich dahingehend zum einen durch flexible Prozessstrukturen aus. Zum anderen entstehen mit den großen Mengen an Prozessen bei GPM im Großen auch große Mengen an Event-Daten, sogenannte „Big Data“. Zur Evaluation dieser Mechanismen werden Log-Files benötigt, in welchen die Charakteristika des GPM im Großen kodiert sind. Dieser Beitrag stellt zunächst einen Ansatz zur Synthetisierung solcher Log-Daten vor, der die Möglichkeit bietet, auf Basis von Prozessarchitekturen Variationen von Logs zu erzeugen, die ggf. Sicherheitseigenschaften verletzen. Zum anderen wird die Implementierung des entwickelten Ansatzes, das Tool *cLog*, gezeigt. Es verwendet spezifische Funktionen von jBPM, einem Java-basierten Open Source Workflow-Management-System, und kann Logs in gängigen XML-basierten Miningformaten (XES, MXML) erzeugen.

1 Einleitung

„GPM im Großen“ dient als Sammelbegriff für die Beschreibung von Ansätzen für das Management komplexer, realer Unternehmensprozesse [17]. Während sich konventionelle GPM-Forschung vor allem auf einzelne Prozesse oder Szenerien mit wenigen isolierten Prozessen konzentriert, liegt die bevorstehende Herausforderung für GPM darin, große Mengen an ineinandergreifenden, zwischenbetrieblichen Prozessen zu bewältigen. Dies gilt auch für Sicherheitsmechanismen, wie Monitore und Audits. GPM im Großen weist unter anderem folgende Eigenschaften auf [17]: zwischenbetriebliche Ökosysteme und Business Networks; flexible Prozessstrukturen; zunehmende Zahl eng miteinander verbundener, voneinander abhängiger Prozesse; konfigurierbare Prozessmodelle, große Prozessportfolien.

Existierende GPM-Methoden und -analysewerkzeuge sind jedoch für GPM im Kleinen konzipiert, d.h. für die Analyse einzelner bzw. kleiner Mengen von Prozessmodellen. Deshalb besteht in der Tat ein Bedarf an Mechanismen, die für die Herausforderungen von GPM im Großen bereit sind (vgl. [17]). Wo groß angelegte Prozesse in (ggf. cloud-basierten) service-orientierten Umgebungen oder gar im globalen Netz ausgeführt werden, können elementare Aktivitäten fein- oder grobkörnigen Protokollereignissen zugewiesen werden. Hierdurch weisen Prozessprotokolle zunehmend alle

typischen Eigenschaften von „Big Data“ auf, wie große physische Verbreitung, Formatvielfalt, nicht-standardisierte Datenmodelle oder heterogene Semantik.

Um die Funktionsweise von Sicherheitsmechanismen, wie Audits [3, 4] oder Monitore [2], testen zu können, werden Ereignisprotokolle (engl. Event-Logs) benötigt, die Prozessausführungen enthalten, welche zum einen Prozessflexibilität, zum anderen Prozessverletzungen (Non-Compliance) einschließen und dabei strukturelle Schwachstellen, Prozessdynamik, beabsichtigte Attacken und Benutzerfehler nachahmen [23, 24]. Derartige Ereignisprotokolle können als Eingabe für Monitoring-, Auditing- und Mining-Werkzeuge dienen und ermöglichen dabei Entwicklern deren Kill-Rate, also die Präzision, mit welcher die Verletzung von festgelegten Sicherheits- oder Compliance-Eigenschaften oder Abweichungen vom Prozessmodel identifiziert wird, zu bewerten. Geschäftsprozesssimulationstechniken können diese Durchläufe generieren. Jedoch gibt es aktuell kein Verfahren (z.B. [37, 9, 35]) oder Tool (z.B. [39, 41, 7, 8], vgl. [18] für eine Übersicht) welches die kontrollierte Generierung von Ereignisprotokollen berücksichtigt und dabei Flexibilität und Sicherheitsverletzungen eines spezifischen Prozesses beachtet [1, 34].

Der hier vorgestellte Beitrag versucht das Verhalten des BPM im Großen hinsichtlich der dabei entstehenden großen Log-Daten zu emulieren. Er lässt sich in einen konzeptionellen Teil und einen praktischen Teil mit anschließender Evaluation der wesentlichen Aspekte gliedern. Im konzeptionellen Teil wird eine Vorgehensweise vorgestellt, mithilfe der die Erzeugung von „Big Log Data“ ermöglicht werden soll. Dies umfasst die Konfiguration von Simulationsläufen, welche GPM Charakteristika, wie bspw. Prozessvariabilität, kodiert, und die symbolische Ausführung von Geschäftsprozess-architekturen. Im praktischen Teil wird das Stand-Alone-Tool *cLog* vorgestellt, welches den entwickelten Ansatz implementiert und große Logs generieren kann. Bei der Implementierung wurden spezifische Funktionen von jBPM, einem Java-basierten Open Source Workflow-Management-System (WFMS) verwendet.

Aufbau: Abschnitt 2 erläutert die für diesen Beitrag als relevant angenommenen Charakteristika von GPM im Großen. Abschnitt 3 stellt den allgemeinen Ansatz vor, dessen wesentliche Bausteine detaillierter in Abschnitt 4 erläutert werden. Das Tool *cLog* wird in Abschnitt 5 beschrieben und hinsichtlich der wesentlichen Aspekte evaluiert.

2 Charakteristika des GPM im Großen

Ziel des Ansatzes ist es, Logs zu generieren, welche einer GPM im Großen Szenerie entstammen könnten. Deshalb werden nur Charakteristika des GPM im Großen (Abbildung 1) in Betracht gezogen, die direkte Auswirkungen auf Logs haben. Dies sind in erster Linie Charakteristika hinsichtlich der Prozessstruktur.

Charakteristik 1, 3, 4, 5, 10 (im Sinne von „lose gekoppelten“ Prozessen) charakterisieren die Prozesskonstellationen, welche bei GPM im Großen auftreten können (von „lose gekoppelt“ bis hin zu großen miteinander verbundenen, voneinander abhängigen Prozessen). Diese Charakteristika spiegeln sich in Geschäftsprozessarchitekturen wider. Eine Geschäftsprozessarchitektur ist eine Sammlung von Geschäftsprozessen und ihren Wechselwirkungen miteinander [11]. Der Fokus liegt also nicht, wie bei Choreographien von Prozessen, auf der Interaktion selbst, oder wie bei Orchestrierungen, auf der zentralen Steuerung eines Prozesses, sondern auf allen Prozessen und den Wechselwirkungen der Prozesse miteinander. [11] beschreibt unterschiedliche Arten von Relationen die zwischen einzelnen Prozessen in einer Prozessarchitektur vorkommen: Komposition (repräsentiert einen Geschäftsprozess

Charakteristika	GPM im Kleinen	GPM im Großen
1. Zeitlicher und räumlicher Fokus der Prozessdefinition	Fokussiert eine oder wenige kooperierende Organisationen	Fokussiert umfassende Geschäftsnetzwerke und Ökosysteme, an denen viele Organisationen teilhaben
2. Strukturierung der Prozesse	Oft a priori festgelegte Prozessstrukturen	Flexible Prozessstrukturen, die an sich ändernde Situationen anzupassen sind
3. Anzahl verschiedener Prozesstypen	Oft a priori festgelegte Anzahl definierter Kern- und Supportprozesse	Wachsende Anzahl von Prozessen durch konfigurierbare Prozessmodelle und große Prozessportfolios
4. Abhängigkeit zwischen Prozessen	Oft wenige Abhängigkeiten	Viele Abhängigkeiten
5. Anzahl der Anspruchsberechtigten	Geringe Anzahl von Anspruchs-berechtigten in einzelnen Prozessen	Hohe Anzahl von Anspruchsberechtigten in unternehmensübergreifenden Prozessen
6. Dynamik der Prozessveränderung	Oft stabile und selten angepasste Prozessstrukturen	Oft angepasste und flexibel konfigurierte Prozessstrukturen
7. Heterogenität verwendeter Modellnotationen	Geringe Heterogenität	Ausgeprägte Heterogenität
8. Anzahl unterschiedlicher Institutionen, die kooperative Prozesse koordinieren	Gering	Hoch
9. Vorherrschende Planungsrichtung	Top-down, oft zentrale Organisation	Bottom-up, dezentrale Organisation
10. Architektur von GPM-Systemen	Oft monolithisch	Lose gekoppelte GPM-Services

Abbildung 1: Charakteristika des GPM im Kleinen und GPM im Großen [17]

als Komposition vieler Subprozesse); Spezialisierung (ein Prozess spezialisiert einen anderen Prozess); Trigger (ein Prozess initialisiert einen anderen Prozess und startet abhängige Prozesse); Informationsfluss (Information oder andere Objekte fließen von einen Prozess zum anderen). Eine Architektur kann entsprechend der Kompositionsrelation nach [25] als Prozessmodell mit Subprozessen betrachtet werden. Ein Prozessmodell mit Subprozessen kann „geglättet“ werden; dies bedeutet, dass alle Unterprozesse in der Ebene des Hauptprozesses aufgelöst werden. Eine Prozessarchitektur ist somit mithilfe eines Prozesses ohne Subprozesse beschreibbar. Umgekehrt kann demnach ein Prozess auch als Prozessarchitektur verstanden werden. Ausgehend von diesem Verständnis wird nachfolgend der Begriff „Prozess“, respektive „Prozessarchitektur“, verwendet.

Aufgrund der Relevanz von Geschäftsprozessarchitekturen bei GPM im Großen sind diese in dem entwickelten Ansatz berücksichtigt. Um den Charakteristika 1, 3, 4, 5 und 10 gerecht zu werden, muss es darüber hinaus möglich sein, mehrere Prozesse bzw. Prozessarchitekturen einer möglichen GPM im Großen Szenerie in einer Log-Datei zu repräsentieren.

Charakteristik 2, 3 und 6 von GPM im Großen betonen die Flexibilität von Prozessen. Flexibilität von Geschäftsprozessen wird nach [20] als die Fähigkeit von Geschäftsprozessen definiert, sowohl mit vorhergesehenen als auch unvorhergesehenen Änderungen des Umfelds, in welchem sie agieren, umgehen zu können. Diese aus den Änderungen resultierenden Variationen spiegeln sich letztlich in den Ereignisprotokollen wider. Aus Sicht der Ereignisprotokolle ist es jedoch nicht relevant, auf welche Art und Weise oder zu welchem Zeitpunkt während der Prozessausführung diese Variationen eingeleitet werden, sondern lediglich, dass Sie im Log repräsentiert sind.

3 Überblick des Ansatzes

Abbildung 2 zeigt einen Überblick des Ansatzes. Das Verfahren nimmt zunächst eine Reihe von Prozessmodellen, welche *Prozessarchitekturen* repräsentieren, und generiert schließlich ein Ereignisprotokoll (*Log)*, welches *Traces* der ausgeführten Prozesse enthält.

Zu Beginn werden die *Simulationsläufe konfiguriert*. Die Konfiguration eines Simulationslaufs enthält eine *Prozessarchitektur,* welche durch ein Prozessmodell definiert wird, Modifikationseinstellungen

und die *Anzahl an Traces*, die der Simulationslauf erstellen soll. *Traces* dokumentieren die ausgeführten Aktivitäten eines Prozessmodells als Reihenfolge von Ereignissen.

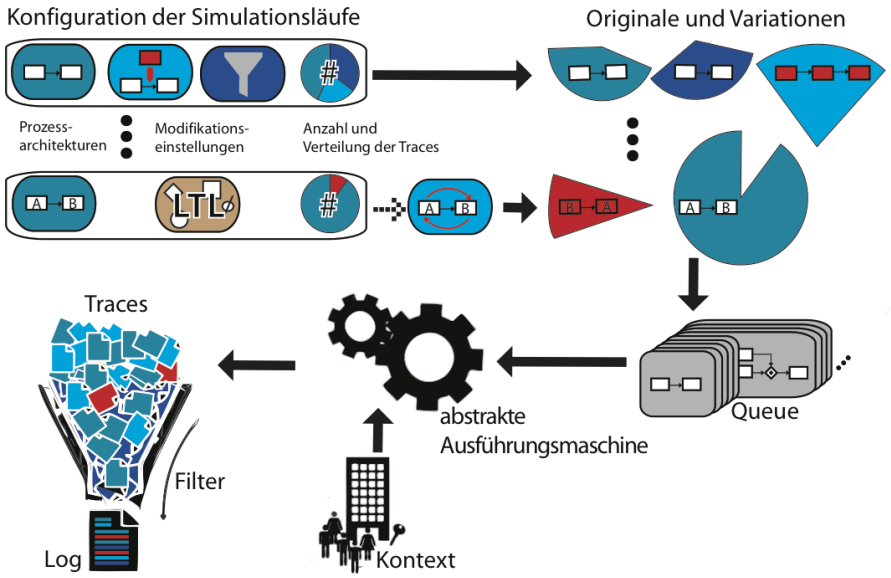

Abbildung 2: Überblick des Ansatzes

Die Modifikationseinstellungen bestimmen den Einsatz von *Transformationsoperatoren* und *Filter*. *Transformationsoperationen* erzeugen Variationen des Prozessmodells vor dessen symbolischer Ausführung. Sie ersetzen bestimmte Kontrollfluss-Muster eines Prozesses nach Weber et al. [30]. *Filter* modifizieren die *Traces* nach der Ausführung. Sie können Prozessaktivitäten hinzufügen oder entfernen und damit übersprungene Aktivitäten oder lückenhaftes Protokollieren nachahmen, sowie die Informationen innerhalb von Traces ändern, sodass geschäftsbedingte Eigenschaften, wie Aufgabentrennung oder Aufgabenbindung (separation- / binding of duties) oder Autorisierungseinschränkungen, durchgesetzt oder verletzt werden. Die Menge an modifizierten Traces wird als Anteil der gesamten *Anzahl der Traces* (siehe Tortendiagramme) konfiguriert.

Die Modifikationseinstellung kann in zweierlei Modi geschehen: Die direkte Konfiguration der Transformationsoperatoren und Filter ohne Kenntnis von Sicherheitsanforderungen an die gegebene Prozessarchitektur (Policy-Unaware Modus), oder die Konfiguration gegebene Sicherheitsanforderungen weiter zu erfüllen oder zu verletzen (Policy-Aware Modus). Um Sicherheitsanforderungen spezifizieren zu können, werden *Compliance-Regeln* verwendet. Zur Durchsetzung einer Verletzung einer Compliance-Regel werden die dazu notwendigen Transformationsoperatoren oder Filter definiert.

Auf Basis der *Konfiguration der Simulationsläufe* werden Variationen der Prozesse generiert. Die *Queue* steuert die Reihenfolge in welcher die *Originale und Variationen* ausgeführt werden. Bei simulierten Prozessen wird angenommen, dass diese in einem Kontext ausgeführt werden. Der Kontext gibt Subjekte an, die autorisiert sind Prozessaktivitäten auszuführen und Objekte, welche von Aktivitäten verwendet werden. Zur Ausführung der Prozesse und der generierten Variationen wird eine *abstrakte Ausführungsmaschine* angenommen, die als Eingabe einen Prozess mit dessen Kontext nimmt und als Ausgabe eine Zeichenkette liefert, welche die Ausführungs-Traces enthält. Schließlich werden alle Traces in einer Log-Datei gespeichert. Die erzeugten Logs werden in unterschiedlichen Formaten ausgegeben, wie Extended Event Stream (XES) oder MXML (traditionelles Format für Prozessmining), sowie Comma Separated Values (CSV).

4 Bausteine

In diesem Abschnitt werden die wesentlichen Bausteine aus Abbildung 2 definiert, die benötigt werden, um die gewünschten Konfigurationsmöglichkeiten zu ermöglichen. Hierzu wird in Abschnitt 4.1 erläutert wie ein Prozess und dessen Ausführungskontext definiert ist. Abschnitt 4.2 präsentiert Operatoren zur Modeltransformation oder Modifikation von Logs. In Abschnitt 4.3 werden die Compliance Regeln erläutert.

4.1 Prozessspezifikationen und Ausführungskontext

Die Hauptbestandteile eines Ereignisprotokolls resultieren zum einen aus der Spezifikation von Prozessen, welche Aktivitäten und deren mögliche Ausführungsreihenfolge definiert, und zum anderen aus dem Ausführungskontext, welcher das organisatorische Umfeld zur Prozessausführung im Sinne von Subjekten definiert, die autorisiert sind Prozessaktivitäten auszuführen und Objekte, welche von Aktivitäten genutzt werden. Für Simulationszwecke wird eine Prozessspezifikation mit ihrem Kontext zu einem Simulationssystem verbunden. Die Spezifikationen basieren dabei auf [34] (enthält formelle Beschreibung).

Die Spezifikation der Prozesse definiert Prozessaktivitäten und die Reihenfolge, in welcher diese ausgeführt werden. Um den Fokus auf die Simulationsprozedur zu legen wird an dieser Stelle von konkreten Spezifikationsmetamodellen abstrahiert. Die einzige Voraussetzung ist, dass die Prozesse wohlstrukturiert sind [30]: Ein wohlstrukturiertes Prozessmodell ist ein Model, das aus Blöcken besteht, die verschachtelt sein können, aber nicht überlappen dürfen. Ein Block kann dabei eine einzelne Aktivität, eine Sequenz, eine parallele oder bedingte Verzweigung oder ein Schleifen-Block sein. Es wird angenommen, dass der Kontrollfluss eines Prozesses mithilfe von OR-, XOR- oder AND-Gatter (implizit oder explizit) definiert wird. Die Gatter definieren die Anzahl der verbundenen ausgehenden Knoten, welche auf herkömmliche Art ausgeführt werden können.

Ein Trace wird als Ausführungssequenz einer Instanz der Prozessspezifikation betrachtet, die eine Menge zeitlich geordneter Ereignisse enthält, von denen jedes von der Ausführung einer Prozessaktivität berichtet. Aktivitäten werden von Subjekten zu einem bestimmten Zeitpunkt ausgeführt und beinhalten verschiedene Objekte. Freigaben beschreiben Rechte von Subjekten, Aktivitäten auszuführen. Als Limitation des Beitrages ist zu erwähnen, dass dies nur eine Klasse von Sicherheitsanforderungen darstellt (siehe Fazit). Andere Konzepte wie Obligationen oder die Verbindung von Objekten mit Aktivitäten selbst, z.B. dass eine bestimmte Ressource nicht zur Ausführung einer Aktivität genutzt werden darf, sind momentan nicht abbildbar. Traces eines Prozesses bezeichnen die Menge aller Ereignisse, die sich auf die Ausführung einer Prozessinstanz beziehen.

4.2 Transformation von Prozessen und Filter

Transformationen betrachten nur den Kontrollfluss, also die Struktur, eines Prozesses und ändern die Reihenfolge, in der Aktivitäten ausgeführt werden können. Prozesstransformationen sind mithilfe von Transformationsfunktionen definiert. Hierbei werden folgende Operatoren auf Gatter-Ebene betrachtet:

- And2Xor: Diese Transformation liefert eine Variation des Eingangsprozesses bei welcher ein bestimmtes AND-Gatter in ein XOR-Gatter umgeschrieben wird. Dies bedeutet: Während der Original-Prozess verlangt, dass beide Teile des AND-Gatters durchlaufen werden, geht die Variante davon aus, dass nur ein (zufällig ausgewählter) Pfad durchquert wird.

- Xor2And: Das Gegenteil von And2Xor bedeutet, dass ein bestimmtes XOR-Gatter in ein AND-Gatter umgeschrieben wird.
- Swap: Diese Transformation liefert eine Variante, bei der die Reihenfolge zweier Aktivitäten vertauscht wird.
- Insert: Eine Aktivität wird hinzugefügt.

Im Gegensatz zur persistenten Änderung des Verhaltens eines Prozesses durch strukturelle Veränderung, können alternativ Filter benutzt werden, um Eigenschaften durchzusetzen oder zu verletzen. Dabei operieren Filter auf gültigen Traces, welche während des Simulationsprozesses generiert wurden. Dies geschieht in nachverarbeitender Art und Weise, bevor sie der Ausgabeprotokolldatei hinzugefügt werden. Ein Filter nimmt einen gültigen Prozess-Trace als Eingabe und generiert eine modifizierte Version dieses Traces entsprechend der angegebenen Parameter, die nicht notwendigerweise gültig ist. Es sollte beachtet werden, dass in Situationen, in denen mehr als ein Filter auf einen Trace angewandt werden, die vom Filter modifizierten Felder „gesperrt“ sein müssen. Dies soll verhindern, dass weitere Filter die durch vorangegangene Anwendungen von Filtern schon durchgesetzten Eigenschaften korrumpieren.

Momentan enthält der Ansatz folgende 6 Filter: **delay** (fügt der Prozessausführung eine Verzögerung hinzu); **skip** (erzeugt Prozessausführungen in welchen manche Aktivitäten übersprungen werden); **silent** (bildet die Situation ab, wenn eine bestimmte Aktivität nicht protokolliert wurde); **authentication** (ahmt eine Zugriffsrecht-Policy und deren Verletzung nach); **binding of duty** (BOD) (imitiert die Einhaltung oder Verletzung einer Binding of Duty Anforderung); und **separation of duties** (kann als Gegenteil zu BOD gesehen werden: Sie besagt dass eine bestimmte Aktivität von einem unterschiedlichen Subjekten ausgeführt werden muss). Aufgrund der Platzbeschränkung wird auf die Definition der Filter in [34] verwiesen. Dabei ist zu erwähnen, dass die Liste der Transformationsoperatoren und Filter erweitert oder hinsichtlich anderer Anwendungsbereiche neu definiert und schließlich *cLog* hinzugefügt werden kann.

4.3 Spezifikation von Sicherheitsanforderungen

Um Sicherheitsanforderungen für eine gegebene Prozessarchitektur spezifizieren zu können, können Compliance-Regeln definiert werden [30, S. 298]. Eine Compliance Regel beschreibt, wie ein interner oder zwischenbetrieblicher Geschäftsprozess entworfen oder ausgeführt werden muss. Hinsichtlich des Kontrollflusses schränken sie die Reihenfolge ein, in welcher Prozessaktivitäten ausgeführt werden können. Eine Compliance Regel kann als Funktion definiert werden, die widerspiegelt ob eine Prozessinstanz - repräsentiert durch deren Ausführungs-Trace - die Regel einhält oder nicht.

Als Voraussetzung zur Überprüfung der Geschäftsprozesscompliance von vorgegebenen Geschäftsprozessen, Prozessinstanzen oder Prozessausführungsprotokollen, müssen die zugehörigen Compliance Regeln in einer maschinenlesbaren Form vorhanden sein. In der Literatur existieren dafür verschiedene Ansätze [5, 6, 10, 22]. Eine beliebte Methode zur Definition und Darstellung von Compliance Regeln ist die Verwendung von linearer temporaler Logik (LTL)[16]. Ramezani et al. [28] hat hinsichtlich des Kontrollflusses aus aktueller Literatur [6, 10, 15, 33] 50 Compliance Regeln zusammengetragen und diese in 15 Kategorien klassifiziert. Diese Kategorien gründen im Wesentlichen auf Dwyers Liste von Patterns, also Mustern, für die Beschreibung von Eigenschaften zur Verifikation endlicher Automaten [10]. Nach Dwyer enthält jede spezifizierte Eigenschaft unter anderem die konkrete Anforderung, das Muster, den Geltungsbereich und ein Mapping. Letzteres stellt eine Abbildung der Eigenschaft auf eine formale Beschreibungssprache dar, wie bspw. LTL.

Auf dieser Grundlage werden in [12, 13] Compliance Patterns und das zugehörige LTL-Mapping definiert [12, Tabelle 2]. Compliance Patterns sind Abstraktionen von häufig benutzten Compliance Einschränkungen (Regeln) und bilden einen Zwischenschritt zur formalen Compliancespezifikation [12]. Diese Patterns bilden die Grundlage für die gewählten Compliance Regeln. Aus Platzgründen werden die gewählten 19 LTL-Formeln an dieser Stelle nicht angeführt. In Abschnitt 6 wird hierzu beispielhaft die Compliance Regel *IFTHENBEFORE(A, B)* erläutert. Als Limitation wird angenommen, dass die durch eine Compliance Regel angegebene Sicherheitsanforderung an einen gegebenen Prozess in selbigem initial erfüllt ist.

5 cLog: Prototypische Implementierung und erste Evaluation

Der Ansatz zur Synthetisierung großer Ereignisprotokolle auf Basis einer flexibel konfigurierbaren Szenerie des GPM im Großen wurde als eigenständige und erweiterbare Java-Anwendung implementiert und basiert auf dem Rahmenwerk, welches in den Abschnitten 3 und 4 geschaffen wurde. Die Anwendung heißt *cLog* („Complex Event Log Synthesis Tool“) und steht unter http://bit.ly/cLogTool zum Download bereit. *cLog* verwendet spezifische Funktionen von jBPM, ein Java-basiertes Workflow-Management-System. jBPM unterstützt eine Untermenge der BPMN2 Spezifikation, was gleichzeitig die Wahl der Modellierungsnotation der Prozesse in *cLog* ist und aufgrund ihrer weiten Verbreitung von Vorteil ist. Abbildung 3 zeigt, wie die jBPM Komponenten in *cLog* integriert und erweitert wurden.

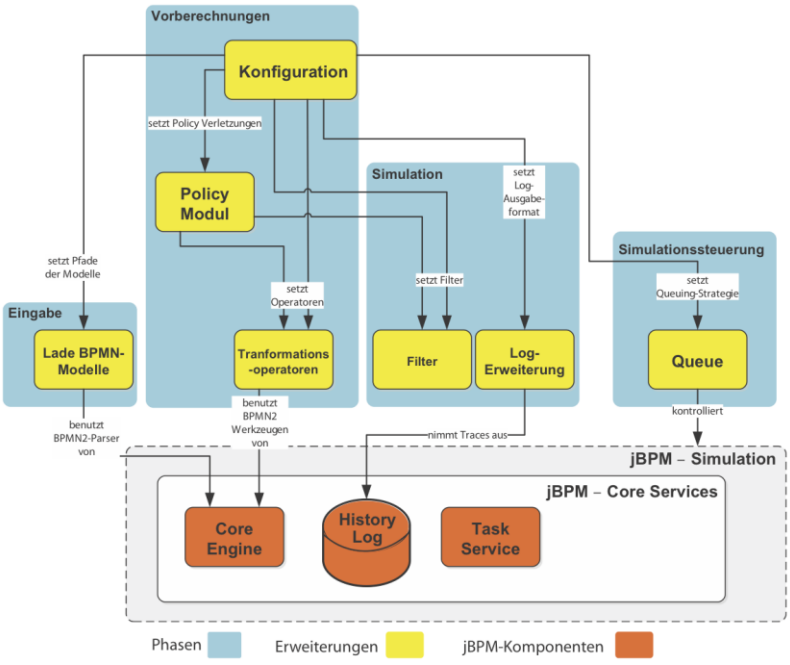

Abbildung 3:jBPM Komponenten und Erweiterung

jBPM bietet mit der Version 5.4 die Möglichkeit zur Simulation. Die jBPM-Simulation bildet eine Erweiterung der jBPM Core-Engine. Bevor die Simulation ausgeführt wird, werden die Aktivitäten in einem Prozess mit „simulierten“ Aktivitäten ersetzt, welche auf Simulationseigenschaften basieren, die im Prozessmodell definiert wurden.

Abbildung 4 zeigt die Konfigurationsansicht (*Configuration*) von *cLog* und die Einstellungsansicht (*Settings*) in welcher die Prozessmodifikationen eingestellt werden. Neben diesen Ansichten gibt es

zwei weitere Ansichten. Die *Queuing*-Ansicht ermöglicht dem Benutzer unterschiedliche Strategien auszuwählen, wie die gegebenen Prozessarchitekturen abgearbeitet werden sollen (FIFO, LIFO, Random). In der *Logging*-Ansicht können Speicherort und Format der Log-Datei angegeben werden.

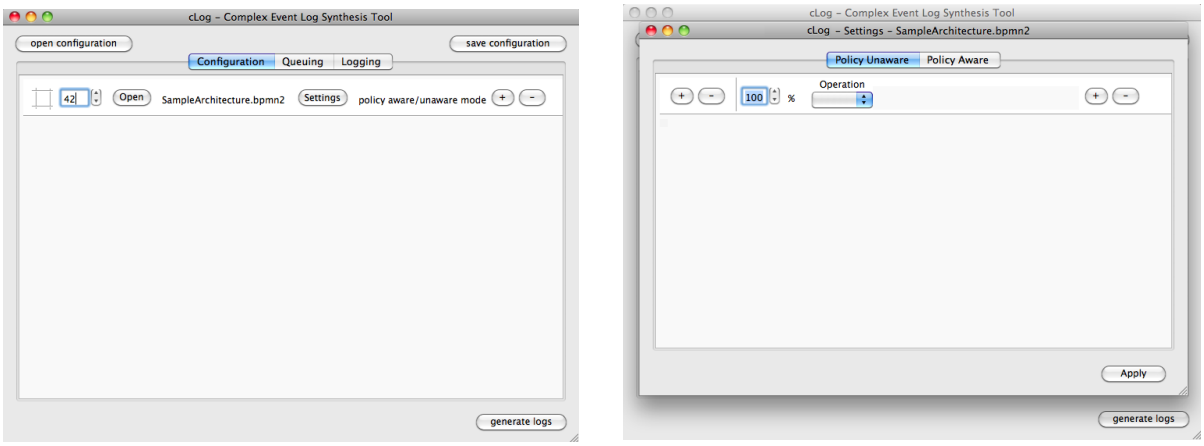

Abbildung 4:Bildschirmfoto der Configuration- und Settings-Ansicht.

Die Hauptansicht des Tools bildet die *Configuration*-Ansicht. Es kann eine Reihe von BPMN-Prozessmodellen geladen und Prozessdurchläufe konfiguriert werden, wodurch eine Simulation der Ausführung von unterschiedlichen Prozessmodellen und unterschiedlichen Konfigurationseinstellungen (*Settings*) ermöglicht wird. Der Benutzer kann in der *Settings-Ansicht* jeder Prozesskonfiguration zwischen einem Policy-Aware oder einem Policy-Unaware Modus wählen (vgl. Abschnitt 3). Im *Policy-Aware* Modus legt der Benutzer fest, welche der vorhandenen Compliance-Regeln verletzt werden sollen. Die Verletzung wird während der Ausführung durch „fest-verdrahteten" Einsatz der definierten Filter und Transformationsoperatoren erzeugt. Im *Policy-Unaware* Modus können direkt die Filter und Transformationsoperatoren eingesetzt werden, wobei je nach gegebenem Prozessmodell deren Einsatz nicht erkennbar ist. Hat der Benutzer schließlich die gewünschte Anzahl an Simulationsläufen für jeden Prozess eingestellt, kann die Synthetisierung beginnen. Alle Konfigurationseinstellungen können in einer XML Datei gespeichert und in *cLog* (bspw. zur Wiederherstellung einer Konfiguration) geladen werden.

5.1 Evaluation

Mit *cLog* wurde eine qualitative und quantitative Evaluation durchgeführt, welche auf die wesentlichen Aspekte fokussiert.

Zur qualitativen Analyse wurden Stichproben verwendet, mit der Annahme, dass die Prozessmodelle die festgelegten strukturellen Einschränkungen einhalten. Die qualitative Evaluation soll zeigen, dass die erzeugten Log-Dateien tatsächlich den Konfigurationen entsprechen, welche vom Nutzer zuvor getätigt wurden. Es wird also überprüft ob die möglichen Verletzungen von Sicherheitseigenschaften, Filter und Transformationsoperatoren effektiv sind. Hierfür wurde ein BPMN-Modell verwendet und die zu testenden Verletzungen der Sicherheitseigenschaften, Filter und Transformationsoperatoren bei der Log Erzeugung angewandt. Zur Überprüfung, ob die Konfigurationsanforderungen erfüllt wurden oder nicht, wurde das Prozessmining-Tool PROM eingesetzt.

In einem Bestellprozess muss zum Beispiel die Aktivität „bestätige Bestellung" *vor* der Aktivität „Bestellabwicklung" ausgeführt worden sein. Diese Sicherheitseigenschaft kann mit der Compliance Regel *IFTHENBEFORE(A, B)* formuliert werden, da diese besagt, dass bei Auftreten der Aktivität A

(„Bestellabwicklung“) Aktivität B („bestätige Bestellung“) vorangegangen sein muss. In der Konfiguration von *cLog* wird diese Eigenschaft für angenommen 10 Prozent der synthetisierten Prozessdurchläufe verletzt, es kommt also zu 10 Prozent zu einer Bestellabwicklung ohne Bestätigung. Nach Erzeugung der Log-Daten konnten anschließend mittels LTL-Checker Plug-In in PROM alle Fälle isoliert werden, welche die Compliance Regel verletzten.

Die quantitative Evaluation betrachtet die Erstellung der Datenmengen und Performanz von *cLog*. Dabei geht es unter anderem darum, tatsächlich große Log-Dateien zu synthetisieren. Zu Beginn der Evaluation mittels *cLog* wurden Log-Daten von 10 GB erzeugt. Durch kombinieren dieser Log-Daten konnten Log-Dateien von 200GB erstellt werden. Bei der Generierung derartig großer Log-Dateien ist außerdem von Bedeutung, wie sich die gewählten unterschiedlichen Formate auf die Performanz auswirken. Messungen der Schreibgeschwindigkeit mit großen Log-Dateien gaben Hinweise darauf, dass die unterschiedliche Schreibperformanz der einzelnen Datei-Formate in etwa den Dateigrößenverhältnissen der unterschiedlichen Mining-Formate entsprechen (MXML hat ca. die doppelte Dateigröße einer XES-Datei (mit gleichen Traces), XES hat die ca. vierfache Größe einer reinen Textdatei).

Darüber hinaus stellt sich hinsichtlich des vorgestellten Ansatzes die Frage, inwiefern dieser tatsächlich GPM im Großen adressiert, da der Ansatz bisher nur auf einzelne Aktivitäten und Freigaben fokussiert und somit sich nicht wesentlich von GPM im Kleinen unterscheidet. Dies liegt in erster Linie an der gewählten Sicht auf Prozessarchitekturen, welche nur im Sinne Ihrer Kompositionsrelation (Prozesse und Unterprozesse) betrachtet werden. Hierdurch wird die Anwendung der in Abschnitt 4 vorgestellten Operatoren ermöglicht, da in diesem Verständnis Prozessarchitekturen auch als Prozesse gesehen werden können. Die Anforderung an den Ansatz war es, Logs zu generieren, die einem GPM im Großen Szenario entstammen könnten. Sicherlich muss dies nicht zwingend der Fall sein. Festzuhalten ist jedoch, dass dieser Beitrag einen ersten Schritt in die Richtung der Berücksichtigung der Charakteristika des GPM im Großen bei der Synthetisierung von Log-Daten darstellt und ein Rahmenwerk schafft, dass hinsichtlich der Repräsentation von GPM im Großen Charakteristika weiter verfeinert werden kann.

6 Verwandte Arbeiten

Geschäftsprozesssimulation ist ein wichtiges GPM-Tool, um die Laufzeit eines Geschäftsprozesses zu prüfen. Simulation kann auch als Mittel genutzt werden Traces zu synthetisieren, um Analysetechniken und -tools zu testen. In beiden Fällen ist es wichtig, Abweichungen von dem vorgeschriebenen Verhalten einfügen zu können um Variabilität zu erreichen (vgl. [36, 38], [21, Kap. 7.4]). Wie in Abschnitt 4 erwähnt, basieren die Konzepte von Transformationsoperatoren und Filter auf [29]. [29] stellt das Tool *SecSy* vor, welches diese Konzepte ebenfalls implementiert. Die wesentlichen Unterschiede dieses Ansatzes und der daraus abgeleiteten Implementierung im Vergleich zu [29] liegen in dem hier gesetzten Fokus auf die Charakteristika des GPM im Großen. SecSy ermöglicht beispielweise sehr feingranulare Simulationseinstellungen in den Prozessen. cLog berücksichtigt hingegen die Ausführungsreihenfolge (Queuing) zwischen den Prozessen. Die Implementierung unterscheidet sich auch darin, dass *SecSy* auf Petri Netzen arbeitet, während cLog BPMN-Modelle verwendet und dazu jBPM integriert. Im Folgenden werden weitere existierende Simulationsansätze und Tools genannt und Unterschiede oder Gemeinsamkeiten mit dem Verfahren dieses Beitrags erläutert.

6.1 Ansätze

Der aktuelle „State of the Art“ berücksichtigt entweder Variationen der Ausführungszeit und des Workflows des Prozesses oder Kontrollflussabweichungen, die Prozessflexibilität und Dynamik nachahmen. [19] stellt einen Simulationsansatz vor, der Dienststörungen simuliert und ihre monetären Auswirkungen auf Geschäftskunden untersucht. Ellouz et al. [14] präsentiert einen Ansatz, der zur Modellierung von Flexibilität eine Versionierung von Prozessmodellen verwendet und damit Simulationsmodelle erstellt. Nakatumba et al. [27] präsentieren einen Ansatz zur Generierung von Ereignisprotokollen mit auslastungsabhängigen Geschwindigkeiten von Simulationsmodellen. Mans et al. [26] verwenden die Simulation, um die Auswirkungen eines zeitplanorientierten Workflow-Management-Systems zu analysieren. Schonenberg et al. [33] entwickeln einen Simulationsansatz für die Analyse von Geschäftsentwicklungen. Rozinat et al. [31] schlagen einen Ansatz für das Entdecken von Simulationsmodellen vor. Diese Arbeiten berücksichtigen Verzögerungen und Auslastungsänderungen, jedoch keine Sicherheit oder Compliance-Regeln.

6.2 Tools

Es gibt einige Tools zur Simulation von Geschäftsprozessen ([18] enthält eine Übersicht von 2006). Das „Alaska Simulator Toolset“ unterstützt unterschiedliche Ansätze um Prozessflexibilität zu berechnen und systematisch zu vergleichen [40]. Bahrami [7] präsentiert ein bei IBM entwickeltes Spezial-Tool, dessen Fokus auf der Unternehmensarchitektur und nicht auf Prozessen liegt. Burattin und Sperduti [8] stellen ein Rahmenwerk für die Generierung von Geschäftsprozessmodellen und ihre Ausführungsprotokolle vor. Jedoch bietet das Tool keine Kontrolle über die generierten Modelle und Ereignisprotokolle. Die AristaFlow BPM Suite unterstützt Simulation. In [20] werden Compliance Erweiterungen von AristaFlow diskutiert und vorgenommen. Die Erweiterungen verfolgen die Idee, durch Einfügen von Kontrollaktivitäten während der Ausführungszeit Compliance zu gewährleisten. Die Verletzung der Compliance wird dabei nicht beleuchtet. Das „CPNTools Toolset“ bietet eine hochgradig konfigurierbare Simulationsumgebung. Jedoch ist das Toolset nicht nur für diesen Zweck entwickelt worden und verlangt für speziellere Anwendungsfälle zunächst weiterer Programmierung.

Im Großen und Ganzen wurde die Generierung von „defekten“ Daten allgemein zur Software-Prozessverbesserung angewandt [29], jedoch nicht im GPM-Bereich. Der Grad der Kontrollierbarkeit ist bei keinem der aufgeführten Tools so, dass man Logs generieren kann, welche einem GPM im Großen Szenario entstammen könnten. Es existiert demnach kein Verfahren oder Tool, das erlaubt die geforderte Flexibilität und das Vorhandensein mehrerer Prozesse, die miteinander interagieren oder unabhängig voneinander sind, zu simulieren und dabei Sicherheitseigenschaften in Betracht zu ziehen, die bei der Simulation verletzt werden können. Diese Arbeit entwickelt ein Verfahren und ein Tool nach diesen Anforderungen und bildet dabei einen Schritt Richtung Erzeugung „defekter“ Daten im GPM-Bereich.

7 Zusammenfassung und Ausblick

Dieses Papier stellt einen Ansatz hinsichtlich der konfigurierbaren, geschäftsprozessbezogenen Synthetisierung großer Log-Daten unter Berücksichtigung des GPM im Großen vor. Der Fokus lag dabei auf der Entwicklung eines Tools, welches diese Synthetisierung ermöglicht.

In Zukunft könnte der Ansatz und schließlich das Tool dahingehend erweitert werden, dass neue Transformationsmöglichkeiten hinzu kommen, die weitere Eigenschaften von GPM in Großen abbilden. In [11] werden Prozessarchitekturen hinsichtlich Ihrer Ereignisse untersucht und ein Ansatz

vorgestellt, der zeigt, wie die unterschiedlichen Ereignis-Relationen kategorisiert und verändert werden können. Somit könnten nicht nur einzelne Aktivitäten und Freigaben, sondern auch Ereignisse zwischen den Instanzen berücksichtigt werden. Darüber hinaus wäre eine Erweiterung der Compliance-Regeln denkbar. Hinsichtlich der Performance bei der Synthetisierung großer Log-Dateien würde sich anbieten ein Parallelisierungsansatz zu entwickeln, der die Berechnungen der Traces verteilt.

8 Literatur

[1] Accorsi, R (2013): A security-aware simulation method for generating business process event logs. In: Accorsi, R; Ceravolo, P; Cudré-Mauroux, P (Hrsg), Proceedings of the 3rd International Symposium on Data-driven Process Discovery and Analysis, Riva del Garda.

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

[3] Accorsi, R; Stocker, T (2012): On the exploitation of process mining for security audits: the conformance checking case. In: Ossowski, S; Lecca; P (Hrsg), Proceedings of the 27th Annual ACM Symposium on Applied Computing. Riva del Garda.

[4] Accorsi, R; Stocker, T; Müller, G (2013): On the exploitation of process mining for security audits: the process discovery case. In: Shin, SY; Maldonado, JC (Hrsg), Proceedings of the 28th Annual ACM Symposium on Applied Computing. Coimbra.

[5] Alberti, M; Chesani, F; Gavanelli, M; Lamma, E; Mello, P; Montali, M; Torroni, P (2008): Expressing and verifying business contracts with abductive logic programming. International Journal of Electronic Commerce 12(4):9–38.

[6] Awad, A., Decker, G., & Weske, M. (2008): Efficient compliance checking using bpmn-q and temporal logic. In: Dumas, M; Reichert, M; Shan, MC (Hrsg), Business Process Management. Springer, Berlin Heidelberg.

[7] Bahrami, A; Sadowski, D; Bahrami, S (1998): Enterprise architecture for business process simulation. In: Medeiros, DJ; Watson, EF; Carson, JS; Manivannan, MS (Hrsg), Proceedings of the 30th conference on Winter simulation. Los Alamitos.

[8] Burattin, A; Sperduti, A (2011): Plg: a framework for the generation of business process models and their execution logs. In: Daniel, F; Barkaoui, K; Dustdar, S (Hrsg), Business Process Management Workshops: BPM 2011 Internationale Workshops. Springer, Berlin Heidelberg.

[9] Cho, YH; Kim, JK; Soung Hie, K (1998): Role-based approach to business process simulation modeling and analysis. Computers & industrial engineering 35(1):343–346.

[10] Dwyer, MB; Avrunin, GS; Corbett, JC (1998): Property specification patterns for finite-state verification. In: Ardis, MA; Atlee, JM (Hrsg), Proceedings of the second workshop on Formal methods in software practice. New York.

[11] Eid-Sabbagh, RH; Dijkman, RM; Weske, M (2012): Business process architecture: Use and correctness. In: Barros, AP; Gal, ABA; Kindler, E (Hrsg), Business Process Management. Springer, Berlin Heidelberg.

[12] Elgammal, A; Türetken, O; van den Heuvel, WJ. Using patterns for the analysis and resolution of compliance violations. International Journal of Cooperative Information Systems 21(1): 31–54.

[13] Elgammal, A; Türetken, O; van den Heuvel, WJ; Papazoglou MP (2010): Root-cause analysis of design-time compliance violations on the basis of property patterns. In: Maglio, PP; Weske, M; Yang, J; Fantinato, M (Hrsg), Proceedings of the 8th International Conference on Service-oriented Computing. San Francisco.

[14] Ellouze, F; Chaâbane, MA; Bouaziz, R; Andonoff, E (2013): Modeling and simulation versions of business process using petri nets. In: Laux, F (Hrsg), The Fifth International Conference on Advances in Databases, Knowledge, and Data Applications. Sevilla.

[15] Fahland, D; De Leoni, M; Van Dongen, BF; Van Der Aalst, WM (2011): Conformance checking of interacting processes with overlapping instances. In: Rinderle-Ma, S; Toumani, F; Wolf, K (Hrsg), Business Process Management. Springer, Berlin Heidelberg.

[16] Gabbay, D; Pnueli, A; Shelah, S; Stavi, J (1980): On the temporal analysis of fairness. In: Abrahams, PW; Lipton, RJ; Bourne SR (Hrsg): Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. New York.

[17] Houy, C; Fettke, P; Loos, P; Van Der Aalst, WMP; Krogstie J (2011): Geschäftsprozessmanagement im Großen. Wirtschaftsinformatik 53(6):377–381.

[18] Jansen-Vullers, M; Netjes, M (2006): Business process simulation – a tool survey. In: Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN. Aarhus.

[19] Kieninger, A; Berghoff, F; Fromm, H; Satzger, G (2013): Simulation-Based Quantification of Business Impacts Caused by Service Incidents. In: e Cunha, JF; Snene, M; Nóvoa, H (Hrsg), Exploring Services Science. Springer, Berlin Heidelberg.

[20] Kittel, K; Sackmann, S; Göser, K (2013): Flexibility and Compliance in Workflow-Systems - The KitCom Prototype. In: Deneckère, R; Proper, HA (Hrsg), Proceedings of the 25th International Conference on Advanced Information Systems Engineering. Valencia.

[21] La Rosa, M; Mendling, J; Reijers, HA (2013): Fundamentals of Business Process Management. Springer, Berlin Heidelberg.

[22] Liu, Y; Muller, S; Xu, K (2007): A static compliance-checking framework for business process models. IBM Systems Journal 46(2):335–361.

[23] Lowis L; Accorsi, R (2009): On a classification approach for soa vulnerabilities. In: Ahamed, SI; Bertino, E; Chang, CK; Getov, V; Liu, L; Ming, H; Subramanyan, R (Hrsg), Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference. Seattle.

[24] Lowis, L; Accorsi, R (2011). Vulnerability analysis in soa-based business processes. IEEE Transactions on Services Computing 4(3):230–242.

[25] Malinova, M; Leopold, H; Mendling, J (2013): An Empirical Investigation on the Design of Process Architectures. In: Alt, R; Franczyk, B (Hrsg), Tagungsband der 11. Internationale Tagung Wirtschaftsinformatik. Leipzig.

[26] Mans, RS; Russell, NC; Van Der Aalst, W; Bakker, PJ; Moleman, AJ (2010): Simulation to Analyze the Impact of a Schedule-aware Workflow Management System. Simulation 86(8-9):519-541.

[27] Nakatumba, J; Westergaard, M; Van Der Aalst, WMP (2012): Generating Event Logs with Workload-Dependent Speeds from Simulation Models. In: Bajec, M; Eder, J (Hrsg), Advanced Information Systems Engineering Workshops. Springer, Berlin Heidelberg.

[28] Ramezani, E; Fahland, D; Van Der Aalst, WMP (2012): Where did I misbehave? diagnostic information in compliance checking. In: Gal, ABA; Kindler, E (Hrsg), Business Process Management. Springer, Berlin Heidelberg.

[29] Raninen, A; Toroi, T; Vainio, H; Ahonen, JJ (2012): Defect data analysis as input for software process improvement. In: Dieste, O; Jedlitschka, A (Hrsg), Proceedings of the 13th International Conference on Product-focused Software Process Improvement. Madrid.

[30] Reichert, M; Weber, B (2012): Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Springer, Berlin Heidelberg.

[31] Rozinat, A; Mans, RS; Song, M; Van Der Aalst, WMP (2009): Discovering simulation models. Information Systems 34(3):305–327.

[32] Said, IB; Chaâbane, MA; Andonoff, E (2010): A model driven engineering approach for modelling versions of business processes using BPMN. In: Abramowicz, W; Tolksdorf, R (Hrsg), Business Information Systems. Springer, Berlin Heidelberg.

[33] Schonenberg, H; Jian, J; Sidorova, N; Van Der Aalst, WMP (2010): Business trend analysis by simulation. In: Pernici, B (Hrsg), Advanced Information Systems Engineering. Springer, Berlin Heidelberg.

[34] Stocker, T; Accorsi, R (2013): SecSy: Security-aware Synthesis of Process Event Logs. In: Jung, R; Reichert M (Hrsg), Proceedings of the 5th International Workshop on Enterprise Modelling and Information Systems Architectures. St. Gallen.

[35] Tumay, K (1996): Business process simulation. In: Charnes, JM; Morrice, DJ; Brunner, DT; Swain, JJ (Hrsg), Proceedings of the 28th Conference on Winter Simulation, Coronado.

[36] Van Der Aalst, WMP (2010): Business process simulation revisited. In: Barjis, J (Hrsg), Enterprise and Organizational Modeling and Simulation. Springer, Berlin Heidelberg.

[37] Van Der Aalst, WMP; Nakatumba, J; Rozinat, A; Russell, N (2010): Business process simulation. In: Brocke, J; Rosemann, M (Hrsg), Handbook on Business Process Management 1. Springer, Berlin Heidelberg.

[38] Vom Brocke, J; Rosemann, M (2010): Handbook on Business Process Management 2: Strategic Alignment, Governance, People and Culture. Springer, Berlin Heidelberg.

[39] Weber, B (2013): Alaska simulator. http://www.alaskasimulator.org. Abgerufen am 19.09.2013.

[40] Weber, B; Pinggera, J; Zugal, S; Wild, W (2010): Alaska simulator toolset for conducting controlled experiments on process flexibility. In: Soffer, P; Proper, E (Hrsg), Information Systems Evolution. Springer, Berlin Heidelberg.

[41] Westergaard, M; Verbeek HMW (2013): CPN Tools. http://cpntools.org/. Abgerufen am 19.09.2013.