Pattern-Mining-Konzepte für die Analyse großer Mengen von Simulationsverlaufsdaten

Niclas Feldkamp[1*], Sören Bergmann[1]

[1]Informationstechnik in Produktion und Logistik, TU Ilmenau, Max-Planck-Ring 12, 98693 Ilmenau, Deutschland; **niclas.feldkamp@tu-ilmenau.de*

Abstract. Üblicherweise basiert die Auswertung der Ergebnisse einer Simulationsstudie auf den Kennzahlen eines abgeschlossenen Simulationslaufs oder mehrerer Simulationsexperimente. Dieser Beitrag präsentiert eine Methode zur Analyse von Simulationsverlaufsdaten, d. h. Daten und Ereignisse, die während der Simulation anfallen. Zu diesem Zweck verwenden wir in diesem Beitrag Pattern-Mining-Algorithmen, um verborgene Zusammenhänge und potenziell interessante Muster in diesen dynamischen Daten zu entdecken. Hierzu wird ein Konzept vorgestellt sowie anhand einer einfachen, akademischen Fallstudie prototypisch angewendet. Diese Methode bietet einen neuen Ansatz zur Erkenntnisgewinnung in komplexen Simulationsszenarien, insbesondere im Bereich Produktion und Logistik.

Einleitung

Die diskret-ereignisgesteuerte Simulation ist ein essenzieller Bestandteil für die Analyse von komplexen Systemen, zum Beispiel im Bereich Produktion und Logistik, aber auch in vielen anderen Anwendungskontexten. Üblicherweise basiert die Auswertung der Ergebnisse einer Simulationsstudie auf den Kennzahlen eines abgeschlossenen Simulationslaufs oder von gleich mehreren abgeschlossenen Simulationsläufen aus verschiedenen Experimenten [1, 2]. Die Auswertung von Simulationsläufen und -experimenten ist in der Simulationsforschung gut durchdrungen, daher gibt es umfassende Literatur hierzu [1, 3–5]. Weniger intensiv erforscht hingegen sind Konzepte für die Auswertung von Daten, die innerhalb eines Simulationslaufs anfallen und durch die zeitliche Dynamik innerhalb eines Simulationslaufs Schwankungen unterliegen. Dies sind in der Regel Daten zu Ereignissen und Werte von Modellzustandsvariablen. In diesem Beitrag verwenden wir hierfür als Oberbegriff den Ausdruck *Simulationsverlaufsdaten*. Am weitesten verbreitet ist hier sicherlich die Trace-File-Analyse, welche die Ereignisliste des Simulators zum Zwecke der Validierung des Modells überprüft [6]. In diesem Beitrag soll jedoch eine Methode vorgestellt werden, mit deren Hilfe versteckte, interessante und potenzielle nützliche Zusammenhänge in Simulationsverlaufsdaten entdeckt werden können. Hierfür verwenden wir das dem Data-Mining-Werkzeugkasten entnommene Pattern Mining. Pattern-Mining-Algorithmen sind in der Lage, Muster und Regeln in großen Mengen von Transaktions-, Log- und Sequenzdaten zu finden und können dabei auch zeitliche Zusammenhänge miteinbeziehen können [7].

Der Aufbau dieses Beitrags ist wie folgt: Im ersten Kapitel werden die Grundlagen zu Simulationsverlaufsdaten gegeben, als auch die Grundlagen zu diversen Vertretern von Methoden aus der Klasse von Pattern-Mining-Algorithmen. Hieran schließt sich eine Übersicht über den Stand der Forschung zum Thema Analyse von Simulationsverlaufsdaten an. In Kapitel 2 wird dann ein Konzept vorgestellt, wie solche Methoden auf Simulationsverlaufsdaten anzuwenden sind und welche Voraussetzungen und Limitationen hierbei zu beachten sind. Dies wird dann in Kapitel 3 anhand einer einfachen, akademischen Fallstudie demonstriert. Kapitel 4 beendet den Beitrag mit einem abschließenden Fazit und einem Ausblick für zukünftige, hieran anzuknüpfende Forschungsmöglichkeiten.

1 Grundlagen und Stand der Forschung

1.1 Simulation und Simulationsverlaufsdaten

Abstrakt betrachtet lösen Ereignisse der diskret-ereignisgesteuerten Simulationstechnik Zustandswechsel zu einem konkreten Zeitpunkt mit einem dazugehörigen Zeit-

stempel aus, wie beispielsweise das Auftreten einer Störung einer Maschine. Das Auftreten von Ereignissen zieht dann in der Regel Änderungen an den Werten der Zustandsvariablen des Simulationsmodells nach sich [8, 9]. Das Ereignis des Eintritts einer Entität in eine Warteschlange erhöht beispielsweise die aktuelle Länge dieser Warteschlange, beispielhaft gezeigt in Abbildung 1.

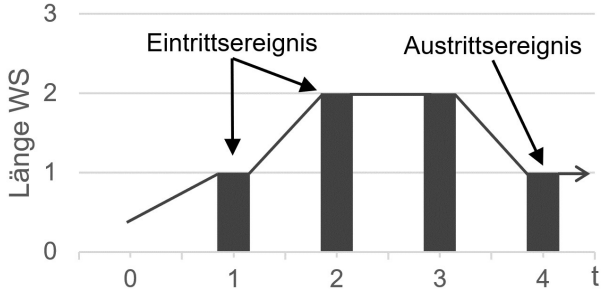


Abbildung 1: Beispiel für den dynamischen Verlauf einer Zustandsvariable.

Derartige Ereignisse und Zustandsänderungen über die Zeit zählen wir in diesem Beitrag ebenfalls zu den Simulationsverlaufsdaten. Das Persistieren von Ereignissen und Zustandsänderung über die Zeit wird in der diskret-ereignisgesteuerten Simulation oftmals auch als Eventlogs oder Tracefiles bezeichnet. Detaillierte konzeptionelle Überlegungen dazu finden sich im Kapitel 2.

1.2 Frequent Pattern Mining und verwandte Methoden

Frequent Pattern Mining ist eine bekannte Data-Mining-Methode und wird klassischerweise für die Warenkorbanalyse genutzt [7]. Hierbei wird nach Artikeln gesucht (sog. Items), die von Kunden häufig zusammen mit bestimmten anderen Artikeln gekauft werden (sog. Frequent Itemsets) [7, 10]. Daraus lassen sich wiederum Regeln generieren, die beschreiben, mit welcher Wahrscheinlichkeit beim Kauf eines Artikels auch ein anderer Artikel üblicherweise gekauft wird. Ziel hierbei ist dann zum Beispiel das Erstellen von gezielten, kundenspezifische Werbemaßnahmen und Empfehlungen. Grundlage hierfür ist das Vorhandsein einer sog. Transaktionsdatenbank. Diese speichert die Transaktionen zu jedem Kunden, wobei eine Transaktion aus einem oder mehreren Items besteht. Aus dieser werden dann algorithmisch die häufigen Itemsets berechnet [7, 10]. Eine davon abgeleitete Methode ist das Sequence Mining [11]. Hierbei wird die Transaktionsdatenbank um einen temporalen Faktor erweitert. Im Szenario der Warenkorbanalyse bedeutet dies, dass ein Kunde, der zu verschiedenen Zeitpunkten Transaktionen durchgeführt hat, somit eine Sequenz von Transaktionen generiert. Diese Daten können dann folglich nach häufigen Sequenzen durchsucht werden, also nach Itemsets, die häufig in einer bestimmten Reihenfolge auftreten [11]. Diese Methode hat neben der Warenkorbanalyse viele weitere Anwendungszwecke, beispielsweise Clickstream-Analysen zur Untersuchung von Webtraffic [12] oder Gen-Analysen im Biomedizinischen Bereich [13]. Ein weiterer, interessanter Vertreter dieser Klasse von Data-Mining-Methoden ist das Periodic Pattern Mining [14]. Im Gegensatz zum Sequential Pattern Mining werden hier nicht mehrere Sequenzen nach häufigen Mustern durchsucht, sondern eine (in der Regel längere) Sequenz nach wiederholt auftretenden Zyklen. Ziel ist es hierbei, die Regelmäßigkeit solcher Zyklen zu bestimmen, wie etwa *"alle 5 Zeiteinheiten tritt Ereignis A auf"*. Dies ist unter anderem hilfreich für die Analyse von Logfiles bei repetitiven Prozessen, beispielsweise im Kontext von Eventlogs bei Internet-of-Things(IoT)-Geräten [14].

1.3 Automatisierte Auswertung von Simulationsverlaufsdaten

Die Idee zur automatisierten Auswertung von Simulationsverlaufsdaten besteht in der Forschung im Bereich von Simulation im Allgemeinen und diskret-ereignisgesteuerter Simulation im Speziellen schon seit geraumer Zeit. So zeigten bereits Tolujev et al. Ansätze zur dynamischen Metamodellierung von Simulationsmodellen auf Basis von Trace-Dateien von GPSS-Modellen auf, zum Beispiel zur Unterstützung bei der Modellvalidierung [15]. Mit dem Aufkommen von leistungsfähigerer Recheninfrastruktur und damit einhergehendem Potenzial für größere Simulationsmodellkomplexität wurden dann auch Data-Mining-Methoden für die Verarbeitung und Auswertung der immer größer werdenden Mengen von Simulationsverlaufsdaten herangezogen. Beispielsweise zeigt Tekinay die automatisierte Analyse von Simulationsverlaufsdaten mittels Pattern Mining und Markov-Ketten für komplexe diskret-ereignisgesteuerte Simulationsmodelle im großen Maßstab auf, mit dem Ziel der automatisierten Modellabstraktion [16]. Eng verknüpft ist hierbei auch das Process-Mining, welches das Erkennen und Analysieren zugrundeliegender Prozesse basierend auf Eventdaten aus Echtsystemen ermöglicht [17], im Gegensatz zu Eventdaten (hier Simulationsverlaufsdaten genannt) aus Simulationsmodellen. Özkul et al. verwenden beispielsweise Process-Mining-Methoden

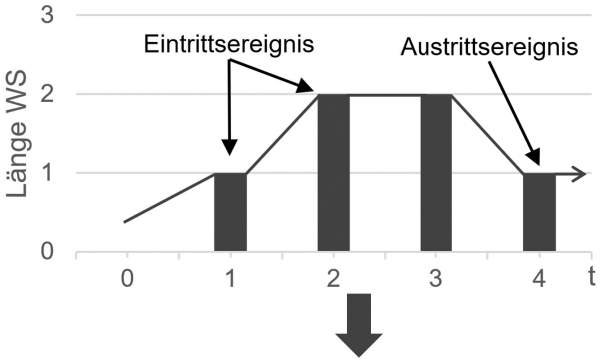
zur Verifikation von Simulationsmodellen [18]. Oftmals findet die Verknüpfung von Simulation und Process Mining allerdings eher in die andere Richtung statt. Aus Echtweltdaten, wie beispielsweise Logfiles aus Realsystemen, werden mit Hilfe von Process-Mining-Methoden Simulationsmodelle generiert, siehe zum Beispiel [19, 20].

Dieser Beitrag fokussiert sich allerdings auf die automatisierte Simulationsverlaufsdatenanalyse mit dem Ziel der Wissensentdeckung mittels Pattern-Mining-Methoden. Feldkamp et al. beschreiben die Wissensentdeckung in Simulationsdaten als Prozess zur Unterstützung der Entscheidungsfindung, indem durch das Aufdecken von unbekannten Zusammenhängen im System potenziell nützliches Wissen generiert wird [3]. Dies bezieht sich primär auf Eingangs- und Ergebnisdaten der Simulation, welche aus mehreren Läufen bzw. Experimenten stammen. Hierbei werden also aggregierte Werte am Ende eines Simulationslaufs betrachtet und nicht der zeitlich-dynamische Verlauf innerhalb eines Laufs. Eine automatisierte Analyse von Simulationsverlaufsdaten mit dem Ziel einer Wissensgenerierung findet sich in nur wenige Arbeiten. Kemper und Tepper befassen sich mit automatisierter Analyse von Simulationsverlaufsdaten für Simulationsmodelle im Kontext von Logistik. Hierzu entwickelten sie eine spezielle Technik zur visuellen Analyse. Ziel ist hierbei das Entdecken von Fehlern oder abnormalem Systemverhalten [6, 21]. Wustmann et al. verwenden Graphenanalyse, um im Kontext von Materialflusssimulationen mögliche Probleme und Ursachen im Ablauf zu identifizieren [22]. Bogon et al. verwenden Sequence-Mining-Methoden um Assoziationsregeln innerhalb eines Simulationslaufs zu entdecken [23]. Allerdings wird hier nicht näher darauf eingegangen, wie aus den Simulationsverlaufsdaten die jeweiligen konkreten Sequenzen für eine Transaktionsdatenbank generiert werden können, um Sequence-Mining-Algorithmen darauf anzuwenden.

Es lässt sich festhalten, dass ein grundsätzliches Konzept für die Anwendung von Frequent-Pattern-Mining-Methoden zur Analyse von Simulationsverlaufsdaten fehlt. Dieses Konzept wird im nächsten Kapitel beschrieben.

2 Konzeptionelle Überlegungen

In diesem Kapitel wird ein allgemeines Konzept zur Nutzung von Pattern-Mining-Methoden für die Analyse von Simulationsverlaufsdaten vorgestellt, basierend auf den drei in Kapitel 1.2 vorgestellten Methoden: Frequent Pattern Mining, Sequential Pattern Mining und Periodic Pattern Mining. Wie bereits in Kapitel 1.1 beschrieben, zählen wir in diesem Beitrag in der diskret-ereignisgesteuerten Simulation zu den Simulationsverlaufsdaten sowohl die auftretenden Ereignisse als auch den Werteverlauf von Zustandsvariablen über die Zeit. Für das Finden von häufigen Mustern und Regeln in diesen Daten mittels Frequent Pattern Mining benötigt man, wie bereits in Kapitel 1.2 dargestellt, eine Transaktionsdatenbank, welche aus Items und Itemsets besteht. Diese Items sind also gleichzusetzen mit den auftretenden Ereignissen in den Simulationsverlaufsdaten (Events). Die (numerischen) Zustandsvariablen müssen hingehen zunächst ebenfalls zu kategorialen Events diskretisiert werden, bevor diese als Items in einer Transaktionsdatenbank genutzt werden können. Abbildung 2 zeigt hierfür exemplarisch ein Beispiel. Im oberen Teil der Abbildung sieht man den Verlauf einer beispielhaften Warteschlange *WS*. In der darunter befindlichen Liste wurde der Zustandsverlauf dieser Warteschlange zu künstlichen, kategorialen Events mit Zeitstempel der Zustandsänderung diskretisiert.



t	Ereignis
1	„Länge WS = 1“
2	„Länge WS = 2“
4	„Länge WS = 1“

Abbildung 2: Diskretisierung von numerischen Zustandsvariablen zu kategorialen Ereignissen.

Hierbei ist zu beachten, dass bei Zustandsvariablen mit sehr großen Wertebereichen die Anzahl der Klassen und deren Wertebereiche für die Diskretisierung möglichst geschickt gewählt werden müssen, sodass nicht zu viele verschiedene Itemsets für die Transaktionsdatenbank generiert werden. Dies könnte sonst dem Auffinden häufiger Muster entgegenstehen. Für eine Warteschlange mit

großer Kapazität könnte man beispielsweise eine Diskretisierung auf Basis vorher definierter, prozentualer Schwellwerte bezogen auf die max. Kapazität wählen, wie etwa in folgendem Beispiel:

```
{"Länge WS ≤ 50 %",
"Länge WS > 50 % & ≤90%", "Länge WS > 90 %"}
```

Hier muss je nach individuellem Anwendungsfall eine passende Diskretisierungslösung gefunden werden. Dies erfordert eventuell weiteres Hintergrundwissen über das zugrunde liegende System sowie eine vorherige Analyse der Werteverteilung, zum Beispiel mittels einer Histogrammvisualisierung. Sind nun sämtliche zu betrachtende Simulationsverlaufsdaten zu kategorialen Ereignissen diskretisiert worden, entsteht hieraus implizit ein Zeitstrahl mit Events entlang der Simulationszeit, wie exemplarisch in Abbildung 3 gezeigt ist.

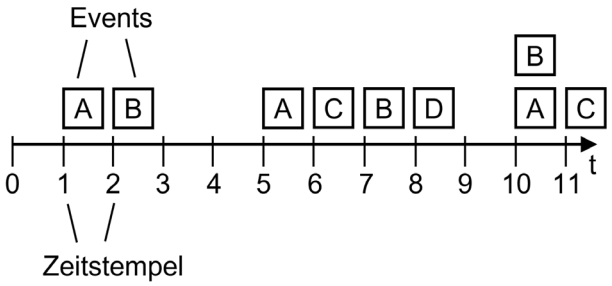


Abbildung 3: Beispielhafter Zeitstrahl mit Events im Simulationsverlauf.

Aus dieser Ereignisliste muss dann die Transaktionsdatenbank für das jeweilige Pattern-Mining-Verfahren generiert werden. Die zentrale Frage ist nun, wie man also aus der Ereignisliste eine Liste mit Transaktionen bzw. Sequenzen erzeugt. Hierbei sind jeweils bestimmte Transformationsregeln und Einschränkungen zu beachten, die sich je nach konkretem Pattern-Mining-Verfahren unterscheiden. Abbildung 4 zeigt dies jeweils exemplarisch anhand der in Abbildung 3 dargestellten Daten. Dies wird nun im Folgenden näher erläutert.

Für das klassische Frequent Pattern Mining (Abbildung 4a) können Events zu einer Transaktion zusammengefasst werden, die innerhalb einer zuvor gewählten, festen Zeitspanne liegen. Für das gewählte Beispiel liegt diese Zeitspanne bei t+2. Der Algorithmus würde hier folglich das häufige (in 80 % der Transaktionen) Muster {A, B} und die Regel *„wenn A, dann B“* finden, welche in 100 % der Fälle zutrifft. Dies ist die einfachste Variante und hat den offensichtlichen Nachteil, dass Muster mit Events, die zeitlich weit auseinander liegen, hierbei nicht gefunden werden.

(a) Frequent Pattern Mining (für t+2)

ID	Events
1	{A, B}
2	{A, C, B}
3	{C, B, D}
4	{D, A, B}
5	{A, B, C}

(b) Sequence Mining (für Startevent A)

ID	Events	Zeit
1	{A}	1
1	{B}	2
2	{A}	5
2	{C}	6
2	{B}	7
2	{D}	8
3	{A, B}	10
3	{C}	11

ID	Sequenz
1	{A}, {B}
2	{A}, {C}, {B}, {D}
3	{A, B}, {C}

(c) Periodic Pattern Mining

S(Event)	Event-Sequenz
S(A)	{1, 5, 10}
S(B)	{2, 7, 10}
S(C)	{6, 11}
S(A, B)	{1, 5, 10)

Abbildung 4: Erzeugen der Transaktionsdatenbanken aus den Simulationsverlaufsdaten für unterschiedliche Frequent-Pattern-Mining-Methoden

Sequence Mining (Abbildung 4b) berücksichtigt auch die zeitliche Dimension bzw. Reihenfolge der Items und Itemsets. Hierbei muss also die Eventliste in mehrere Sequenzen zerlegt werden. Hierfür ist ein Startevent zu definieren, welches den Beginn einer neuen Sequenz markiert. Dies kann etwa ein für den Anwender interessantes Event sein, wie beispielsweise der Ausfall einer Maschine. Alle in der Folge aufgetretenen Events werden dann zu der Sequenz hinzugezählt. Im gezeigten Beispiel wurde hier das Event A als Startevent gewählt. Der Al-

gorithmus würde hier die häufige Subsequenz {A, B} finden, selbst wenn A und B zeitlich weiter auseinander liegen. Nachteil hier ist jedoch, dass es eventuell schwierig ist, ein adäquates Startevent auszuwählen und damit eine sinnvolle Sequenzliste zu generieren.

Beim Periodic Pattern Mining (Abbildung 4c) ist das Erzeugen der Transaktionsdatenbank recht simpel zu bewerkstelligen, indem zu jedem Ereignistyp die zeitliche Sequenz aus den dazugehörigen Zeitstempeln gebildet wird. Innerhalb dieser Zeitstempelsequenzen kann der Algorithmus nun nach häufigen bzw. regelmäßigen Zyklen suchen

Periodic Pattern Mining stellt somit die beste Methode zum Analysieren insbesondere großer Mengen von Simulationsverlaufsdaten dar. Die Transaktionsdatenbank kann hier sofort und automatisiert erzeugt werden und es müssen keine Entscheidungen über notwendige Parametrisierungsoptionen bei der Erzeugung der Transaktionsdatenbank getroffen werden. Eventuell kann es aber sinnvoll sein, eine Auswahl der in die Transaktionsdaten zu überführenden Ereignistypen zu treffen, um uninteressante Eventtypen vorab herauszufiltern.

3 Fallstudie

3.1 Aufbau

Im Folgenden wird eine simple, akademische Fallstudie als Veranschaulichung der zuvor diskutierten Methodik präsentiert. Das Simulationsmodell wurde in Anylogic implementiert und besteht aus einem Warteschlangensystem mit zwei Linien, wie Abbildung 5 zeigt.

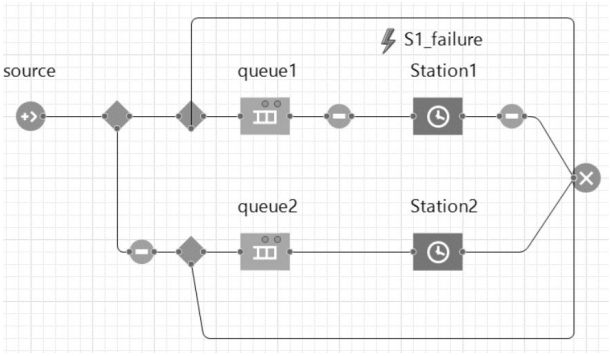


Abbildung 5: Screenshot des Simulationsmodells.

Konkret trägt die obere Bearbeitungsstation *Station1* die Hauptlast des Systems und wird über deren Warteschlange *queue1* mit Aufträgen versorgt. Im Falle einer Störung auf dieser Bearbeitungsstation wird der untere Pfad geöffnet, sodass Aufträge auf die untere Bearbeitungsstation *Station2* umgeleitet werden, bis der obere Pfad wieder zur Verfügung steht. Das Modell wurde als nicht-terminierendes System über einen längeren Zeitraum simuliert und alle auftretenden Events wurden über die Eventdatenbank von Anylogic mitgeschnitten. Dabei wurden neben den Standardereignissen der diskret-ereignisgesteuerten Simulation, wie etwa der Eintritt einer Entität oder das Auftreten einer Störung der Station, einige zusätzliche Ereignisse definiert und entsprechend geloggt, welche für die Auswertung und Analyse potenziell interessant sein könnten. Diese beziehen sich auf Ergebnisgrößen der Simulation und tauchen ohne vorherige manuelle Definition nicht notwendigerweise in der Ereignisliste des Simulators auf, wie beispielsweise eine voll- oder leergelaufene Warteschlange. Abbildung 6 zeigt exemplarisch einen Auszug des Eventlogs.

Main | events_log

	event_name	date	agent_type	agent
1	q2_empty	11-03-2024 00:00:00	Main	root
2	q1_empty	11-03-2024 00:00:00	Main	root
3	source_arrival	11-03-2024 00:00:18	Main	root
4	q1_empty	11-03-2024 00:00:18	Main	root
5	agent_exited	11-03-2024 00:01:18	Main	root
6	source_arrival	11-03-2024 00:01:53	Main	root
7	q1_empty	11-03-2024 00:01:53	Main	root
8	source_arrival	11-03-2024 00:02:59	Main	root
9	q1_empty	11-03-2024 00:02:59	Main	root
10	source_arrival	11-03-2024 00:03:01	Main	root
11	q1_empty	11-03-2024 00:03:59	Main	root
12	source_arrival	11-03-2024 00:08:06	Main	root
13	q1_empty	11-03-2024 00:08:06	Main	root
14	source_arrival	11-03-2024 00:08:09	Main	root
15	source_arrival	11-03-2024 00:08:12	Main	root
16	source_arrival	11-03-2024 00:08:16	Main	root
17	source_arrival	11-03-2024 00:08:19	Main	root
18	source_arrival	11-03-2024 00:09:15	Main	root
19	source_arrival	11-03-2024 00:10:18	Main	root
20	source_arrival	11-03-2024 00:11:32	Main	root
21	source_arrival	11-03-2024 00:12:13	Main	root
22	source_arrival	11-03-2024 00:14:22	Main	root

Abbildung 6: Beispielhafter Auszug aus dem Eventlog des Simulationsmodells.

3.2 Analyse

Zur Analyse der Simulationsverlaufsdaten verwenden wir in dieser Fallstudie ausschließlich die Methode des Periodic Pattern Mining, da diese, wie im Kapitel zuvor beschrieben, die vielversprechendste und interessanteste Methode zum Auffinden von Mustern in Simulationsverlaufsdaten darstellt. Im ersten Schritt wurde hierfür der

zuvor erzeugte Eventlog in ein allgemein kompatibles CSV-Format exportiert. Eine weitere Datenbereinigung ist nicht notwendig, da Simulationsdaten, im Gegensatz zu Echtdaten, fehlerfrei sind, sofern vorausgesetzt werden kann, dass das zugrunde liegende Simulationsmodell valide und fehlerfrei ist. Die Analyse wurde mit Hilfe der Python-Bibliothek Scikit-mine[1] durchgeführt, welche eine schnelle und effiziente API für Pattern Mining in Python bereitstellt. Konkret wurden hier insbesondere die Methoden der Klasse *PeriodicPatternMiner* genutzt.

Wie bereits zu erwarten war, findet der Algorithmus eine Vielzahl von Mustern, von denen für die Auswertung im Sinne einer Wissensentdeckung durch interessante Muster nur wenige tatsächlich relevant sind. Es benötigt also einen gewissen Grad an Hintergrundwissen über das Modell sowie ein manuelles Filtern und Suchen nach Mustern bzw. einzelnen Komponenten und Schlüsselereignissen, um letztendlich interessante Muster aufzudecken. Tabelle 1 zeigt hierzu vier ausgewählte Muster, welche im Folgenden näher erläutert werden.

Nr.	Muster	Abstand
1	(S1_failure)	3:30:32
2	(S1_failure ⇒ [d=0:30:00] ⇒ S1_repair_done)	3:29:48
3	(q2_empty ⇒ [d=0:12:00] ⇒ q2_full)	16:00:01
4	(q2_empty ⇒ [d=0:12:00] ⇒ q2_full ⇒ [d=0:17:00] ⇒ S1_repair_done ⇒ [d=0:46:00] ⇒ q2_empty)	3:30:00

Tabelle 1: Ausgewählte periodische Muster in den Simulationsverlaufsdaten. Der Wert *d* in den eckigen Klammern zeigt jeweils den durchschnittlichen Abstand zwischen zwei Teilkomponenten eines Gesamtmusters an.

Muster 1 zeigt den periodischen Verlauf des Störevents von Station 1 mit einem regelmäßigen Auftreten von 3,5 Stunden im Schnitt. Dies ist nicht überraschend, da die zugrunde gelegte Zufallsverteilung für das Auftreten der Störung mit einem Erwartungswert von 3,5 Stunden parametrisiert war. Dennoch soll dieses Muster als anekdotischer Beweis dienen, dass die gewählte Methodik in der Lage ist, solche erwarteten Regelmäßigkeiten auch tatsächlich zu finden. Als Nebeneffekt kann zudem die vorgestellte Methodik auch zur Überprüfung und Validierung des Modells herangezogen werden. Ein ähnlicher Zusammenhang findet sich in Muster 2, welches sich auf die Periodizität von Störungsauftritt und Reparatur bezieht. Der Wert *d* in den eckigen Klammern zeigt jeweils den durchschnittlichen Abstand zwischen zwei Teilkomponenten eines Gesamtmusters an. Da die im Simulator hinterlegte Entstörungszeit 30 Minuten beträgt, findet der Algorithmus auch hier korrekterweise einen regelmäßigen Zyklus aus Auftreten einer Störung sowie deren Behebung mit einem Abstand von 30 Minuten zwischen beiden Ereignissen und einer Wiederholung von im Schnitt ebenfalls wieder ca. 3,5 Stunden. Muster 3 besteht aus Ereignissen, die sich auf eine Ergebnisgröße im Verlauf beziehen, nämlich den Zustand der Warteschlange *q2*. Wie man sieht, gibt es hier eine Regelmäßigkeit zwischen Leerstand und dem Volllaufen der Warteschlange, wobei der Abstand *d* beider Ereignisse in diesem Muster im Schnitt 12 Minuten beträgt und das Muster mit einer Regelmäßigkeit von 16 Stunden auftritt. Daraus lässt sich schließen, dass bei Öffnen des unteren Pfades nach Auftritt einer Störung auf dem oberen Pfad es regelmäßig 12 Minuten dauert, bis die untere Warteschlange vollgelaufen ist. Muster 4 beschreibt schlussendlich den Zyklus zwischen dem Volllaufen der vorher leeren Warteschlange q2, dem Abschluss der Reparatur von Station 1 sowie das darauffolgende Leerlaufen derselben Warteschlange, mit den entsprechenden Zwischenabstandszeiten der jeweiligen Einzelereignisse. Daraus lässt sich also ableiten, dass nach Behebung der Störung von Station 1 die vollgelaufene Warteschlange q2 regelmäßig nach im Schnitt 46 Minuten wieder leergezogen worden ist.

Es fällt auf, dass Muster 3 einen deutlich größeren Abstand hat, also seltener auftritt, als Muster 4, obwohl die ersten beiden Ereigniskomponenten dieses Musters identisch sind. Dies liegt daran, dass in einigen wenigen Fällen zwar die Regelmäßigkeit für die ersten beiden Ereigniskomponenten von Muster 4 gegeben waren, aber nicht für die beiden hinteren, weshalb daraus in der Folge ein neues Muster (Muster 3) gebildet wurde. Solche Zusammenhänge gilt es bei der Auswertung solcher Muster auf Simulationsverlaufsdaten zu berücksichtigen. Grundsätzlich ist dies eine wesentliche Herausforderung bei der Anwendung von Periodic-Pattern-Mining, die auch in diversen prototypischen Experimenten zu diesem Beitrag festgestellt wurde: Je stärker ein stochastischer Einfluss

[1] siehe https://github.com/scikit-mine/scikit-mine

auf bestimmte Ereignisse wirkt, desto schwieriger wird es, hierzu noch Regelmäßigkeiten zu entdecken. Hierbei werden dann oftmals mehrere zyklische Muster entdeckt, die aber eigentlich derselben kausalen Ereigniskette zuzuordnen wären. Eine Abhilfe könnte hier sein, durch das geschickte Zusammenfassen mehrerer Muster hieraus wieder das originäre zyklische Muster zu errechnen.

4 Fazit und Ausblick

In diesem Beitrag wurde gezeigt, wie mit Hilfe von Pattern-Mining-Methoden Simulationsverlaufsdaten zur Wissensgenerierung verwendet werden können. Hierzu wurden diverse Pattern-Mining-Methoden vorgestellt sowie ein Konzept zur Anwendung dieser auf Simulationsverlaufsdaten eingeführt. Darüber hinaus wurde die Nutzbarkeit von Periodic Pattern Mining im Speziellen exemplarisch anhand einer einfachen, akademischen Fallstudie aufgezeigt. Allerdings verbleiben hierbei noch einige Herausforderungen und daraus abgeleiteter Forschungsbedarf. Eine große Herausforderung besteht darin, dass das Vorhandensein vieler und/oder starker stochastischer Einflüsse im Modell dem Entdecken von zyklischen Mustern entgegenstehen. Abweichungen aufgrund der Stochastik können hier möglicherweise vorhandene Muster und Regelmäßigkeiten überdecken. Eine weitere Herausforderung stellt die schiere Größe von Logfiles und damit einhergehende Masse von Ereignissen in komplexen, langlaufenden Simulationsmodellen dar. Die Grenzen der Anwendbarkeit der hier vorgestellten Methode müssen in weiteren Forschungsarbeiten noch näher untersucht werden, z. B. hinsichtlich der Komplexität des untersuchten Szenarios und des zugrundeliegenden Modells. Hierbei kann es schwierig sein, interessante, nicht-triviale Muster herauszufiltern. Es besteht weiterer Forschungsbedarf für z. B. graphische, interaktive Methoden, die den Anwender hierbei unterstützen. Ein weiteres zukünftiges Forschungsfeld könnte die Erweiterung der Methode für die Nutzung mit Daten aus mehreren Simulationsexperimenten sein. In diesem Beitrag wurde nur ein Szenario mit fixen Faktorwerten betrachtet. Wenn man nun im Sinne eines Experimentdesigns die Werte bestimmter Faktoren verändert, erhält man als Ergebnis einen Simulationsverlaufsdatensatz pro Experiment. Das Auswerten von Mustern und deren Veränderung in Abhängigkeit der Faktorwerte könnte wiederum interessantes und potenziell nützliches Wissen über das Modell hervorbringen.

References

[1] Kelton, W. D. 1995. A tutorial on design and analysis of simulation experiments. In *Proceedings of the 1995 Winter Simulation Conference*, 24–31. DOI=10.1109/WSC.1995.478700.

[2] Law, A. M. 2014. *Simulation Modeling and Analysis*. McGraw-Hill Series in Industrial Engineering and Management Science. McGraw Hill Book Co, New York, N.Y.

[3] Feldkamp, N., Bergmann, S., and Strassburger, S. 2020. Knowledge Discovery in Simulation Data. *ACM Trans. Model. Comput. Simul.* 30, 4, 1–25.

[4] Sanchez, S. M. 2014. Simulation Experiments: Better Data, Not Just Big Data. In *Proceedings of the 2014 Winter Simulation Conference*. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey, 805–816.

[5] Kleijnen, J. P., Sanchez, S. M., Lucas, T. W., and Cioppa, T. M. 2005. State-of-the-Art Review: A User's Guide to the Brave New World of Designing Simulation Experiments. *INFORMS Journal on Computing* 17, 3, 263–289.

[6] Kemper, P. and Tepper, C. 2005. Trace Based Analysis of Process Interaction Models. In *Proceedings of the 2005 Winter Simulation Conference*. IEEE inc., Piscataway, N.J.

[7] Aggarwal, C. C. 2014. An Introduction to Frequent Pattern Mining. In *Frequent Pattern Mining*, C. C. Aggarwal and J. Han, Eds. Springer International Publishing, Cham, Heidelberg, New York, Dordrecht, London, 1–17.

[8] Schriber, T. J., Brunner, D. T., and Smith, J. S. 2014. Inside discrete-event simulation software: How it works and why it matters. In *Proceedings of the 2014 Winter Simulation Conference*. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey, 132–146.

[9] Pedgen, D. C. 2010. Advanced tutorial: Overview of simulation world views. In *Proceedings of the 2010 Winter Simulation Conference*. IEEE, Piscataway, NJ, 210–215.

[10] Han, J. and Kamber, M. 2006. *Data mining*. *Concepts and techniques*. The Morgan Kaufmann series in data management systems. Elsevier; Morgan Kaufmann, Amsterdam, Boston, San Francisco, CA.

[11] Agrawal, R. and Srikant, R. 1993. Mining Sequential Patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*. IEEE inc., Piscataway, N.J., 3–14.

[12] Liu, Z., Wang, Y., Dontcheva, M., Hoffman, M., Walker, S., and Wilson, A. 2017. Patterns and Sequences: Interactive Exploration of Clickstreams to Understand Common Visitor Paths. *IEEE Transactions on Visualization and Computer Graphics* 23, 1, 321–330.

[13] Cellier, P., Charnois, T., Plantevit, M., Rigotti, C., Crémilleux, B., Gandrillon, O., Kléma, J., and Manguin, J.-L. 2015. Sequential pattern mining for discovering gene interactions and their contextual information from biomedical texts. *Journal of biomedical semantics* 6, 27.

[14] Galbrun, E., Cellier, P., Tatti, N., Termier, A., and Crémilleux, B. 2019. Mining Periodic Patterns with a MDL Criterion. In *Machine Learning and Knowledge Discovery in Databases*, M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley and G. Ifrim, Eds. Lecture Notes in Computer Science. Springer International Publishing, Cham, 535–551. DOI=10.1007/978-3-030-10928-8_32.

[15] Tolujev, J., Lorenz, P., Beier, D., and Schriber, T. J. 1998. Assessment of simulation models based on trace-file analysis: a metamodeling approach. In *Proceedings of the 1998 Winter Simulation Conference. Simulation in the 21st century*. IEEE, Piscataway, N.J., 443–450.

[16] Tekinay, Ç. 2022. *Automated Abstraction of Discrete-Event Simulation Models using State-Trace Data*. Doctoral Thesis, Delft University of Technology.

[17] van der Aalst, W. M. P. 2011. *Process Mining*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[18] Özkul, F., Sutherland, R., Wenzel, S., and Spieckermann, S. 2023. Einsatz von Process-Mining zur Verifikation und Validierung von Simulationsmodellen in Produktion und Logistik. In *Simulation in Produktion und Logistik 2023*. ASIM-Mitteilung Nr. 187. Universitätsverlag Ilmenau, Ilmenau, 463–472.

[19] Spieckermann, S., Stöhr, N., Mayer, G., Özkul, F., and Wenzel, S. 2023. Fallbeispiele aus Produktion und Logistik für die Verknüpfung von ereignisdiskreter Simulation und Process-Mining. In *Simulation in Produktion und Logistik 2023*. ASIM-Mitteilung Nr. 187. Universitätsverlag Ilmenau, Ilmenau, 155–165.

[20] Jadrić, M., Pašalić, I. N., and Ćukušić, M. 2020. Process Mining Contributions to Discrete-event Simulation Modelling. *Business Systems Research Journal* 11, 2, 51–72.

[21] Kemper, P. and Tepper, C. 2009. Automated Trace Analysis of Discrete-Event System Models. *IIEEE Trans. Software Eng.* 35, 2, 195–208.

[22] Wustmann, D., Vasyutynskyy, V., and Thorsten, S. 2009. Ansätze zur automatischen Analyse und Diagnose von komplexen Materialflusssytemen. In *5. Fachkolloquium der Wissenschaftlichen Gesellschaft für Technische Logistik*, W. M. Scheid, Ed. Universitäsverlag, Ilmenau, 1–20.

[23] Bogon, T., Timm, I. J., Jessen, U., Schmitz, M., Wenzel, S., Lattner, A. D., Paraskevopoulos, D., and Spieckermann, S. 2012. Towards assisted input and output data analysis in manufacturing simulation: The EDASim approach. In *Proceedings of the 2012 Winter Simulation Conference (WSC 2012)*. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.