Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 2211

Ein Ansatz zur Einführung von Complex Event Processing zum workfloworientierten Software-Monitoring

Sebastian Niehaus[1]

Abstract: In vollautomatisierten Geschäftsworkflows sollte nicht nur die Ausführung, sondern auch die Überwachung der ausführenden Software automatisiert ablaufen. Bei dieser Überwachung bleiben meist diejenigen Ausfälle unberücksichtigt, bei denen die jeweilige Software zwar noch als aktiver Prozess läuft und Ressourcen beansprucht, aber nicht mehr dem eigentlichen Zweck nachgeht. Ein solcher Ausfall wird erst deutlich, wenn man den gesamten Workflow betrachtet. Im vorliegenden Beitrag wird anhand des Auftragseingangsprozesses eines Unternehmens aus der Fleischwarenindustrie eine Möglichkeit vorgestellt, diese Art von Ausfällen mittels Complex Event Processing (CEP) zu erkennen. Dabei wird die Zielsetzung verfolgt, ein ressourcensparendes Echtzeit-Verfügbarkeits-Monitoring für den Auftragseingangsworkflow zu schaffen.

Keywords: Software-Monitoring, Complex Event Processing, Geschäftsworkflow

1 Einleitung und Problemstellung

In automatisierten Geschäftsworkflows ist der fehlerfreie Workflowablauf abhängig von der Verfügbarkeit der genutzten Softwaremodule. Zur Gewährleistung dieser Verfügbarkeit ist es notwendig jeden Ausfall der Software sofort zu erkennen. Im vorliegenden Beitrag wird ein Unternehmen betrachtet, welches in der Fleischwarenindustrie tätig ist und daher besonders auf fehlerfreie Workflowabläufe angewiesen ist. Zu dem Unternehmen gehören über zehn Schlacht- und Verarbeitungsbetriebe, die in ganz Deutschland verteilt sind, um möglichst kurze Lieferzeiten zu ermöglichen. Von diesen Standorten werden die Produkte direkt zu den Kunden geliefert. Da besonders in der Produktion von frischer Ware die zu produzierenden Artikelzahlen täglich bedarfsgerecht bestimmt werden müssen, besteht für Kunden die Möglichkeit, die Bestellung EDI-basiert zu übertragen. Diese Bestellungen werden vollkommen automatisiert verarbeitet und an den entsprechenden Produktionsstandort übermittelt, damit die Produktion und die Lieferung schnellstmöglich erfolgen können. Dem Kunden ist es somit möglich, die Bestellung rund um die Uhr aufzugeben. Die eingegangenen Kundenaufträge werden anschließend sofort bearbeitet und als Produktionsaufträge an die jeweiligen Produktionsstandorte übertragen.

Fällt in diesem Prozess beispielsweise die Software zur Standortzuordnung der

[1] Hochschule Osnabrück, Fakultät Wirtschafts- und Sozialwissenschaften, Landwehrstraße 7a, 49393 Lohne, Sebastian.Niehaus@hs-osnabrueck.de

Kundenaufträge aus, hat das zur Folge, dass ein oder mehrere Aufträge nicht rechtzeitig in der Produktion ankommen und somit auch nicht bedient werden können. Um einem solchen Ausfall entgegenzuwirken bzw. ihn rechtzeitig erkennen zu können, läuft der Prozess auf hochverfügbarer Hardware ab, die über ein entsprechendes Monitoring-System permanent überwacht wird. Darüber hinaus sendet jede bearbeitende Software eine Fehlermeldung, sofern ein Prozessschritt nicht ordnungsgemäß ausgeführt werden kann. Es treten aber immer wieder Ausfälle auf, bei denen Softwaremodule nicht mehr ihrem ursprünglichen Zweck nachgehen und dementsprechend auch keine Meldung mehr über Erfolg oder Misserfolg der Aktivität senden. Dennoch nehmen sie als aktiver Prozess Hardwareressourcen in Anspruch. Ausfälle dieser Art werden meist erst mit Verzögerung, aufgrund von Hinweisen von Anwendern, erkannt und erfordern auch eine manuelle Kontrolle. Diese späte Erkennung ist darauf zurückzuführen, dass in dem betrachteten Beispielprozess gegenwärtig nur ein Softwareinternes Monitoring sowie ein Infrastruktur-Monitoring eingesetzt werden. In [Hei15] beschreibt Hein ebenfalls, dass durch eine reine Infrastruktur-Überwachung Softwareausfälle in den meisten Fällen erst spät erkannt werden. Darüber hinaus werden die Notwendigkeit und der Wert des Monitorings auf Softwareebene aufgezeigt.

2 Vorgehensweise und Zielsetzung

Die Zielsetzung der Arbeit besteht darin, eine Software zum Monitoring des eingangs beschriebenen Workflows zu konzipieren und deren Einführung zu beschreiben. Das Ziel des Monitorings ist aber nicht die Leistungsüberwachung, wie es beispielsweise beim Business Activity Monitoring (BAM) der Fall ist[Sch13]. Die Software soll lediglich inaktive Softwaremodule erkennen und demnach eine reine Verfügbarkeitsüberwachung leisten. Es soll aber nicht nur die Möglichkeit bestehen, eine Aussage über die Verfügbarkeit proaktiver Softwaremodule zu treffen, sondern auch über Softwaremodule, die nicht zur Datenübertragung dienen. Während der gesamten Einführung wird stets darauf geachtet, die Lösung möglichst leicht übertragbar auf andere Prozesse zu gestalten und somit ein Referenzmodell für ähnliche Projekte zu bieten. Daher soll das Monitoring-System so konzipiert sein, dass nach Möglichkeit keinerlei Modifikationen an den ausführenden Softwaremodulen vorgenommen werden und außerdem keine Eingriffe in den Prozessablauf nötig sind.

Daraus wird folgende Forschungsfrage abgeleitet:

Wie lässt sich Complex Event Processing (CEP) zum workfloworientierten Software-Monitoring einsetzen und welche Vorteile ergeben sich aus dieser Art des Software-Monitorings?

Im Folgenden wird zunächst das Complex Event Processing erklärt und anschließend das entwickelte Konzept der Begrenzung des CEP auf Workflows beschrieben. Darauf aufbauend wird die Einführung des Complex Event Processing erläutert und anhand von ausgewählten Szenarien die Herausforderungen bei der Einführung in die Praxis

beschrieben. Zuletzt werden ein Fazit sowie ein Ausblick auf mögliche aufbauende Projekte geboten. Als vergleichbare Arbeiten sind an dieser Stelle [HG09] und [Hal07] zu nennen, die zwar BAM in der Praxis umsetzen und somit leistungsorientiertes Monitoring als primäres Ziel verfolgen. Dennoch können beide Arbeiten ebenfalls als Referenzmodelle für die Einführung von CEP in Geschäftsprozessen betrachtet werden. In [Hei15] wird zur Softwareüberwachung eine Nutzersimulation vorgestellt. Bei dieser Art der Softwareüberwachung findet ebenfalls eine Prozessorientierung statt.

3 Complex Event Processing (CEP)

Complex Event Processing ist eine Softwaretechnologie, die in dem Buch „The Power of Events“ von David Luckham erstmals beschrieben wird[Luc02]. „Allerdings hat CEP viele unabhängige Wurzeln in der Forschung, wie beispielsweise die ereignisorientierte Simulation über aktive Datenbanken und Netzwerkmanagement bis zum temporalen Schließen in der KI“ [EB09].

Complex Event Processing hat zum Ziel, Ereignisse (Events), die miteinander in Verbindung stehen, in Echtzeit zu überwachen. Ein Ereignis kann dabei alles sein, was passiert oder was passieren kann [LS11]. Im Allgemeinen bezieht sich ein Ereignis auf die Veränderungen eines Zustands, also auf Änderungen des Wertes einer Eigenschaft eines realen oder virtuellen Objekts. Wird ein Wert über einen Zeitraum betrachtet, ergeben sich daher Ereignisströme. In diesen Ereignisströmen bestehen Zusammenhänge zwischen den Ereignissen der unterschiedlichen Ströme. In welcher Art sie im Zusammenhang stehen, wird in Ereignismustern abgebildet. Im Wesentlichen wird zwischen einfachen Ereignismustern und komplexen Ereignismustern unterschieden. Einfache Ereignismuster enthalten dabei Ereignisse und boolesche Operatoren, die die Verbindung zwischen den einzelnen Ereignissen aufzeigen. Oftmals reichen diese einfachen Ereignismuster zur Darstellung der Zusammenhänge nicht aus. In diesem Fall müssen weitere Operatoren zur Darstellung herangezogen werden[Alv10]. Mit Hilfe dieser Ereignismuster lassen sich Ereignisregeln formulieren. Eine Ereignisregel beschreibt eine spezifische Reaktion, die beim Erkennen eines Musters ausgeführt wird. Sie besteht aus einem Bedingungsteil und einem Aktionsteil. Der Bedingungsteil enthält ein oder mehrere Ereignismuster und der Aktionsteil enthält eine Aktion oder Reaktion, die beim Auftreten des Musters ausgeführt werden soll [BD15, S.9ff.]. In der Abb.1 wird ein Ereignismuster dargestellt, welches als

$$(A \wedge B \wedge C) \rightarrow D \rightarrow E \rightarrow (G \wedge F)$$

zu interpretieren ist. Dabei legen die Sequenzoperatoren (→) die zeitliche Reihenfolge fest, ebenfalls werden boolesche Operatoren verwendet, wie in diesem Fall das logische Und (∧). Jede Variable (in der formalen Darstellung) und jeder Punkt (in der Grafik) stellen ein Ereignis dar. Für jedes dargestellte Ereignis wäre zu einem anderen Zeitpunkt

auch ein anderes Ereignis eines beliebigen Ereignistyps, also jede andere Änderung eines beliebigen Wertes, möglich.

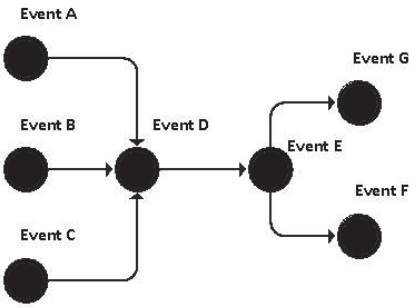

Abb. 1: Ereignisbeziehungen

In der Literatur finden sich zahlreiche Übersichten über kommerzielle und Opensource-CEP-Tools. Bruns und Dunkel nennen eine kleine Auswahl dieser Tools [BD15, S.46] und verweisen zudem für weitere Toolübersichten auf Vidačković et.al. und Vincent. Beide bieten eine sehr umfassende Übersicht und beschreiben zudem die Eigenschaften der Tools grob [VRR10], [Vin14]. In [Rob10] bietet Robins neben einer Übersicht über CEP-Tools, einige Hinweise zur Implementierung. Der Markt für CEP-Lösungen wächst sehr rasant und es besteht ein immer größerer Bedarf an CEP-Lösungen in der Praxis[Lea09].

4 CEP mit Workflowbegrenzung

Für das workfloworientierte Software-Monitoring soll beim CEP nicht mehr jedes Ereignis betrachtet werden, welches mit einem anderen Ereignis in Beziehung steht. Es sollen lediglich diejenigen Ereignisse der Ereignistypen betrachtet werden, die dem zu überwachenden Geschäftsworkflow zugeordnet werden können. Ziel dieser reduzierten Betrachtung soll die Komplexitätsreduzierung und Problemorientierung sein.

Wird für das Beispiel aus Abb.1 unterstellt, dass jedes Ereignis einen anderen Ereignistypen hat und die Ereignistypen der Ereignisse C und G für einen zu betrachtenden Workflow irrelevant sind, können sie ignoriert werden. Damit reduziert sich die Betrachtungsmenge um zwei Ereignistypen, womit ebenfalls die Reduzierung der jeweiligen Ereignisse dieses Ereignistyps einhergeht (Abb. 2). Formal reduziert sich das Ereignismuster damit auf:

$$(A \wedge B) \rightarrow D \rightarrow E \rightarrow F$$

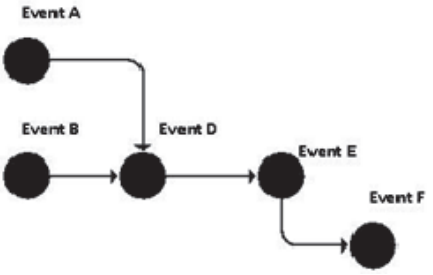

Abb. 2: Reduzierte Sicht auf Eventbeziehungen

Damit ist die Komplexität des Ereignismusters sichtlich verringert worden. Da die Software im Geschäftsworkflow auf Verfügbarkeit überwacht werden sollen, werden keine Geschäftsereignisse betrachtet, sondern rein technische Ereignisse. Für jedes technische Ereignis müssen Sensoren eingesetzt werden, was in diesem Fall Softwaremodule zur Ereigniserfassung sind. Das könnten beispielsweise Module zur Prüfung von Änderungen in Dateiverzeichnissen sein. Jeder Sensor erfasst dabei nur Ereignisse eines Ereignistyps. Die Sensoren werden so gesetzt, dass Ereignismuster entstehen, die eine Aussage über die Verfügbarkeit der Software erlauben. Dabei sollte für die Messwerte jedes Sensors bestimmt werden, inwieweit sie mit den anderen Werten korrelieren und somit relevant für das Ereignismuster sind. Es sollten allerdings nicht nur Korrelationen zwischen einzelnen Ereignissen verschiedener Ereignistypen untersucht werden, sondern auch die Korrelation zu Ereignismengen. Sensoren, die irrelevante Ereignisse erfassen, können entfernt werden.

5 Systemeinführung

Die Einführung von workfloworientiertem CEP zum Software-Monitoring setzt explizites Wissen über den Geschäftsprozess und den verwendeten Ressourcen voraus. Dieses Wissen ist essenziell für die Identifizierung der Ereignisse bzw. zunächst für die Eventtypen. Zur Generierung dieses Wissens wird vor der Systemeinführung zunächst die Durchführung der Prozessanalyse beschrieben und anschließend die Identifikation der Ereignisse.

5.1 Geschäftsprozessanalyse

Aufgrund der Zielsetzung Software zu überwachen, werden diese in der Geschäftsprozessmodellierung als Ausgangspunkt genutzt. Dabei wird der Ansatz der subjektorientierten Geschäftsprozessmodellierung verfolgt, die im Wesentlichen aus zwei Schritten besteht[SFG09]. Im ersten Schritt werden alle Subjekte und ihre

Interaktionsbeziehungen identifiziert. Das heißt, dass die beteiligten Softwaremodule und deren Nachrichtenaustausch identifiziert werden. Dabei sollten nicht nur die Softwaremodule, sondern genau der Client und der Server bestimmt werden. Im zweiten Schritt wird das Subjektverhalten spezifiziert. Das bedeutet, dass jedes Subjekt mittels der Abbildung seines Verhaltens verfeinert wird. Dabei werden streng sequenziell die Zustände und Zustandsübergänge der einzelnen ausgeführten Aktivität beschrieben. Damit werden Geschäftsereignisse identifiziert, die allerdings nicht für die Verfügbarkeitsüberwachung der Software genutzt werden. Zur Unterstützung der Einhaltung der subjektorientierten Modellierungsmethode wird das Modellierungstool Nautilus verwendet, das durch einen Formulierungsassistenten diese Methodik unterstützt. Für die Modellierung wird die eEPK-Notation genutzt, da diese eine umfassende Sicht auf den Geschäftsprozess bietet, welche für die Identifizierung der Messpunkte der technischen Ereignisse hilfreich ist. Allerdings muss zusätzlich für jede Inputentität und jede Outputentität im Model ergänzt werden, wo die Entität abgelegt ist bzw. wie die Entität weiter übertragen wird.

5.2 Identifikation der technischen Ereignisse und Ereignismuster

Für die Identifikation der technischen Ereignisse werden die Geschäftsaktivität und die angebundenen Modellelemente im Geschäftsprozessmodel betrachtet. Es wird für jede Geschäftsaktivität ein Messpunkt identifiziert, von dem aus eine Aussage über die Ausführung getroffen werden kann. Außerdem müssen Messpunkte identifiziert werden, deren Ereignisse eine Verbindung zu den Ereignissen des Aktivitätsmesspunkts aufweisen. Dafür werden auch vorhergehende Aktivitäten betrachtet. Diese Identifizierung wird anhand von zwei Szenarien beschrieben, die sich abstrahiert auf alle anderen Prozessszenarien des Beispielprozesses übertragen lassen:

- Szenario 1: Eine Software zur Konvertierung von Bestellungen liest Dateien verschiedener Datentypen beim Auslösen eines Zeittriggers aus einem Importverzeichnis und konvertiert diese in ein einheitliches Format. Anschließend werden die Dateien in das Exportverzeichnis der Software verschoben.
- Szenario 2: Eine Software zur Leistungsüberwachung prüft, ob alle Dateien ordnungsgemäß konvertiert werden. Bei der Software handelt es sich um keine proaktive Software. Jede Datei in dem Exportverzeichnis wird geöffnet und geprüft, aber nicht verändert oder weiter verarbeitet. Nach einem festdefinierten Zeitpunkt werden die Dateien aus dem Exportverzeichnis geladen und weiterverarbeitet.

Für Szenario 1 lassen sich zwei Messpunkte identifizieren, die gemeinsam betrachtet eine Aussage über die Verfügbarkeit der Software erlauben. Wird die Anzahl der Dateien im Import- und Exportverzeichnis überwacht, sollten die Änderungen in den Verzeichnissen immer ungefähr gleich sein. Damit bestehen zwei Ereignistypen, die miteinander in Verbindung stehen und deren Ereignisse somit als komplexe Ereignisse betrachtet werden können. Die Anzahl im Exportverzeichnis kann minimal abweichen,

da sich im Importverzeichnis Dateitypen befinden könnten, die nicht verarbeitet werden können. Diese Abweichung sollte allerdings für das Verfügbarkeits-Monitoring außer Acht gelassen werden. In der Tab.1 sind Messwerte enthalten, die genau diesen Sachverhalt abbilden. Zum Zeitpunkt T1 und T3 werden alle Dateien konvertiert und zum Zeitpunkt T2 wird eine Datei nicht konvertiert. Würde die Anzahl der Dateien im Exportverzeichnis stark von der Anzahl der Dateien im Importverzeichnis abweichen oder dem Wert 0 entsprechen, ist die Software oder eine Komponente der Software nicht mehr lauffähig. Demnach sind alle Ereignismuster zu identifizieren, bei denen die Änderung des Messwerts im Importverzeichnis stark von der Änderung des Messwerts im Exportverzeichnis abweicht. Für die Darstellung in den Tabellen bleibt unberücksichtigt, dass die Anzahl im Importverzeichnis und im Exportverzeichnis nicht zum selben Zeitpunkt gemessen wird, sondern die Messung der Dateianzahl im Exportverzeichnis erst nach der Konvertierung erfolgt.

Messzeitpunkt	Importverzeichnis	Exportverzeichnis
T1	71	71
T2	64	63
T3	73	73

Tab.1: Messungen für Szenario 1

Alternativ könnten auch Datenbanktabellen oder Dateigrößen überwacht werden. Dabei müssen ggf. Änderungen der Dateigröße bei Bearbeitungsprozessen einbezogen werden. Bei Softwaremodulen in denen Dateien oder andere Datenobjekte nicht abgelegt, sondern direkt weiter übertragen werden, können auch alternative Übertragungswege überwacht werden.

Für Szenario 2 lassen sich keine Daten- oder Dateibewegungen überwachen, allerdings ist die Hardwareressourcennutzung einer solchen Software abhängig von der Anzahl der Dateien. Durch diese Korrelation lässt sich auch die Veränderung der Ressourcennutzung als komplexes Ereignis betrachten. Allerdings müssen bei der Messung des Ressourcenverbrauchs die Ereignisse über einen definierten Zeitraum betrachtet werden. Daher erfolgt auf einem Ereignis bei den Messwerten im Exportverzeichnis eine Vielzahl von Ereignissen bei den Messwerten der Ressourcennutzung. In der Tab.2 ist dieser Sachverhalt abgebildet. Zum Zeitpunkt T1 ergibt sich ein Ereignis an beiden Messpunkten. An nachfolgenden Messzeitpunkten nur noch bei der RAM-Nutzung. Zur Ermittlung besserer Korrelationswerte sollten mehrere Ressourcen einbezogen werden. Sollte die Software nicht mehr das Exportverzeichnis analysieren, würde die Ressourcennutzung einen anderen Verlauf aufweisen. Eine Meldung müsste demnach dann erfolgen, wenn die Klasse von Ereignissen aus der RAM-Nutzung einen für die Anzahl an Dateien untypischen Verlauf hat.

Messzeitpunkt	Exportverzeichnis	RAM-Nutzung
T1	71	3814
T2	71	3867
T3	71	3854

Tab.2: Messungen für Szenario 2

5.3 Umsetzung in der Software

Für die Erfassung der Ereignisse müssen Softwaremodule eingerichtet werden, die als Sensoren agieren. Da im Beispielprozess alle Softwaremodulen auf Microsoft Windows-Betriebssystemen ablaufen, werden für die Softwaremodule PowerShell-Skripte eingesetzt. Um die PowerShell-Skripte nicht nur auf dem lokalen Rechner einzusetzen, werden WMI-Klassen eingesetzt, die in den PowerShell-Skripten verwendet werden. Außerdem werden Befehle aus den Cmdlets verwendet, die einen direkten Eingriff in bestehende Softwaremodule ermöglichen und somit auch Aussagen über einzelne Komponenten liefern. Jedes Skript agiert als ein Sensor, der an nur einem Ereignispunkt eingesetzt wird. Ein Sensor ist dabei eine Ausgabeschnittstelle oder Anwendungscodeerweiterung zur Datenerfassung. Er gibt beim Aufruf beispielsweise die Anzahl der aktuell im Verzeichnis vorhandenen Dateien zurück. Zur Analyse und zur anschließenden Meldung bei Ausfällen wird kein herkömmliches CEP-Tool eingesetzt, da es sich nicht ausreichend an die Prozesse und die Problemstellung anpassen lässt. Stattdessen wird der PRTG Network Monitor eingesetzt, da sich in diesem die Weiterverarbeitung der Rückgabewerte individuell gestalten lässt[Tim15]. Die Skripte werden mit dem Skriptsensor in das Tool eingebunden und anschließend wird im Formelsensor der Zusammenhang abgebildet. Der Skriptsensor und der Formelsensor sind Möglichkeiten zur individuellen Datendarstellung im PRTG Network Monitor. Dabei lassen sich mit dem Skriptsensor externe Datenquellen einbinden und mit dem Formelsensor mehrere Datenquellen zusammenfassen und Berechnungen durchführen. Jede Software ist somit in einem Formelsensor abgebildet und die unterschiedlichen Datenquellen, die somit auch eine Ereignisquelle sind, werden zu einer Aussage zusammengefasst. Zur besseren Übersicht sollten alle Sensoren in einem Prozesssensor vereint werden. Damit lassen sich alle zu überwachenden Applikationen, die dem gleichen Prozess zugeordnet werden können, schneller erfassen. Die Abb. 3 zeigt den Prozesssensor des Beispielprozesses. In dem Prozesssensor sind der Konverter (Szenario 1) und das Filetransfersystem (Szenario 2) eingebunden. Ein Ausfall einer Software würde zu einer Statusänderung im Prozess führen.

Abb. 3: Auftragseingangsprozess im PRTG Network Monitor

6 Kritische Reflektion und Ausblick

Wird CEP workfloworientiert zum Verfügbarkeits-Monitoring eingesetzt, lassen sich Softwareausfälle und sogar einzelne inaktive Softwarekomponenten identifizieren. Außerdem lässt sich, durch eine Beschränkung auf geschäftskritische Prozesse, ein Monitoring realisieren, was auf geschäftskritische Software beschränkt ist. Diese Konzentration ermöglicht die Einsparung von Ressourcen.

Allerdings setzt die Erkennung eines Ausfalls einer Software oder einer einzelnen Softwarekomponente einen nahezu perfekten Zusammenhang zwischen den Ereignissen der unterschiedlichen Messpunkte voraus. Daher stellen Verzeichnisse, die durch mehrere Softwaremodule unterschiedlicher Prozesse genutzt werden, eine Herausforderung dar. Ebenfalls ergeben sich Probleme bei der Zuordnung von Messwerten der Hardwareressourcennutzung, sofern Softwaremodule mehrmals ablaufen. Eine weitere Schwierigkeit besteht in der Fehlermeldung bei Ereignismustern, die außerhalb des erwarteten Wertebereichs liegen, aber dennoch keinen Störung der Software als Ursprung haben. Diesem Problem könnte in einer möglichen Erweiterung der vorgestellten Methodik durch Event stream processing (ESP) entgegen gewirkt werden. Zudem muss für die Inbetriebnahme des implementierten Konzepts die Fehlerbehandlung und Fehlerzuständigkeit genau geregelt werden. In dem Zusammenhang sollte auch untersucht werden, in welchem Maße sich die Fehlerbehandlung automatisieren lässt, damit der vollautomatisierte Workflowablauf gewährleistet werden kann. Darüber hinaus wäre eine Evaluation zur Aufdeckung möglicher Chancen und Risiken des beschriebenen Konzepts sinnvoll und ein Vergleich zu anderen Softwareüberwachungskonzepten aufzuzeigen. Ein weiteres Folgeprojekt wäre die Identifikation von Prozessen im Unternehmen, auf die dieses Konzept übertragen werden kann. Dabei könnte auch untersucht werden, in welchem Umfang eine Übertragbarkeit auf automatisierte Produktionsprozesse oder nur teilautomatisierte Prozesse möglich und nötig ist.

Danksagung

Ein besonderer Dank gilt Prof. Dr. Andreas Schmidt, der mich nicht nur bei der Umsetzung des vorgestellten Projekts, sondern auch bei dem Verfassen dieses Beitrages tatkräftig unterstützt hat. Dieser Beitrag entstand im Rahmen eines Praxisprojekts im Bachelorstudiengang Betriebliches Informationsmanagement an der Hochschule Osnabrück.

Literaturverzeichnis

[BD15] Bruns, R.; Dunkel, J.: Complex Event Processing. Komplexe Analyse von massiven Datenströmen mit CEP, Wiesbaden: Springer Vieweg, 2015.

[EB09] Eckert, M.; Bry, F.: Complex Event Processing (CEP), Informatik_Spektrum_32_2, S.163-167, 2009.

[Hal07] Hoontae, K.; Yong-Han, L.; Hongsoon, Y., Nam Wook C.: Design and Implementation of a Personalized Business Activity Monitoring System. In (Julie A. J. Hrsg.): Human-Computer Interaction. HCI Applications and Services, Berlin, Heidelberg: Springer Verlag, 2007.

[Hei15] Hein, M.: Ende-zu-Ende-Monitoring. Durchblick verschaffen, IT Administrator Sonderheft II/2015, S.68-70,2015.

[HG09] Heinz, G.; Greiner, T.: Business Activity Monitoring mit Stream Mining am Fallbeispiel TeamBank AG, HMD – Praxis der Wirtschaftsinformatik Volume 268, S.82-89, 2009.

[Lea09] Leavitt, N.: Complex-Event Processing Poised for Growth, IEEE Computer Society: Computer, S. 17-20, 2009.

[LS11] Luckham, D.; Schulte, R. et al: Event Processing Glossary – Version 2.0, Event Processing Technical Society, http://www.complexevents.com/wp-content/uploads/2011/08/ EPTS_Event_Processing_Glossary_v2.pdf, Stand: 18.04.2016.

[Luc02] Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise, Addison-Wesley, 2002.

[Rob10] Robins, D. B.: Complex Event Processing, Advanced Topics in Software Systems (CSEP 504), 2010.

[Sch13] Schmidt, W.: Business Activity Monitoring (BAM), In (Rausch, P et al.): Business Intelligence and Performance Management, London: Springer-Verlag, 2013.

[SFG09] Schmidt, W.; Fleischmann, A.; Gilber, O.: Subjektorientiertes Geschäftsprozessmanagement, HMD – Praxis der Wirtschaftsinformatik Volume 266, S.52-62, 2009.

[Tim15] Timmerman, T.: Marktübersicht Monitoring. Von Äpfel und Birnen, IT Administrator Sonderheft II/2015, S.29-35,2015.

[Vin14] Vincent, P.: CEP tooling market survey 2014. http://www.complexevents.com/ 2014/12/03/cep-tooling-market-survey-2014/ (2014). Stand: 18.04.2016.

[VRR10] Vidackovic, K., Renner, T., Rex, S.: Marktübersicht Real-Time Monitoring Software. Event Processing Tools im Überblick, Stuttgart: Fraunhofer Verlag, 2010