2011

Flexibilität in Business Process Management Systemen durch Case-based Reasoning

Andreas Pichler
pichler.andreas@gw-world.com

Flexibilität in Business Process Management Systemen durch Case-based Reasoning

Andreas Pichler
Gebrüder Weiss GmbH
Friedrich-Schindler-Straße 12
A-6922 Kennelbach
+43 / 5574 / 696 - 1404
pichler.andreas@gw-world.com

ZUSAMMENFASSUNG

Business Process Management Systeme sind in der Lage standardisierte Prozesse mit einer Vielzahl an Prozessinstanzen weitgehend autonom auszuführen und zu überwachen. Den Prozessen zugrunde liegen definierte Prozessmodelle, in denen in einer Prozessmodellierungssprache festgelegt ist, wie der Prozessablauf im Einzelnen aussieht. Da es selten möglich ist, alle denkbaren Ereignisse in einem Prozessmodell festzuhalten, kann der so definierte Prozess auf manche Ereignisse nicht in adäquater Weise reagieren. Andererseits existieren mit dem Supply Chain Event Management und dem Case-based Reasoning Methoden, um fallbasiert auf Ereignisse reagieren zu können.

Die vorliegende Arbeit kombiniert diese Konzepte so, dass sich daraus ein Architekturmodell und ein Ablaufmodell für ein dynamisches Business Process Management System ergibt. Dieses System verwendet als Basis standardisierte Prozesse und steuert mittels Case-based Reasoning die Behandlung auftretender Ereignisse. Für den internen Ablauf eines solchen Systems dient das 4R-Modell des Case-based Reasoning als Basis und wird zu einem 6R-Modell erweitert.

Schlüsselwörter

Business Process Management, Business Process Management Systeme, Supply Chain Event Management, Case-based Reasoning.

1. EINLEITUNG

Die Entwicklung von Informationssystemen war und ist in den vergangenen Jahren durch drei eindeutige Trends gekennzeichnet. Die Programmierung von Applikationen wird zunehmend abgelöst durch die Assemblierung von vorgefertigten Modulen zu immer neuen Applikationen. Die Datenorientierung von Informationssystemen wird zunehmend ersetzt durch eine Prozessorientierung. Und zuletzt entfernt sich die Softwareentwicklung immer mehr von einem umfänglichen, antizipativen Design und bewegt sich hin zu einem organischen Wachstum und kontinuierlichen Redesign von Informationssystemen. [1] [2] [3]

Diesen Trends wird im Besonderen durch die Entwicklung von Business Process Management Systemen (BPMS) Rechnung getragen, die sich der betrieblichen Prozesse bewusst sind und deren Ausführung unterstützen. Sie lassen sich daher definieren als generische Softwaresysteme, die von expliziten Prozessdesigns angetrieben werden um operationale Unternehmensprozesse auszuführen und zu verwalten. [1]

Für viele Unternehmen ist die reine Prozessverwaltung, Ausführung und Überwachung allerdings nicht ausreichend. Vielmehr ergibt sich immer öfter die Forderung nach einer schnellen Anpassbarkeit von Prozessinstanzen durch berechtigte Anwender, der möglichen Abweichung von vordefinierten Prozessmodellen und der einfachen Adaption der zugrunde liegenden Prozessmodelle [4]. Dies führt zwangsläufig zu einem Dilemma in der Konzeption von BPMS. Einerseits sollen standardisierte Prozessmodelle als Basis für die Prozessausführung herangezogen werden. Andererseits sollen die Prozessinstanzen nach Bedarf an besondere Anforderungen angepasst werden können. Daraus sollen zugleich Rückschlüsse auf eine generelle Adaption der zugrunde liegenden Prozessmodelle abgeleitet werden [5].

Die vorliegende Arbeit zeigt auf, dass sich dieses Dilemma durch die Kombination eines BPMS mit einer Case-based Reasoning (CBR) Engine und die Anwendung des Konzepts des Supply Chain Event Managements (SCEM) lösen lässt. Das BPMS verfügt über definierte Prozessmodelle und kann diese durch Instanziierung eines Prozessmodells ausführen. Über das SCEM werden Ereignisse, die während der Ausführung einer Prozessinstanz auftreten, erkannt und weiter behandelt. Die Prozessinstanz, das zugrunde liegende Prozessmodell und das aufgetretene Ereignis dienen dann als Basis, um in der Falldatenbank der CBR-Software nach vergleichbaren Fällen aus der Vergangenheit zu suchen, die damals angewandten Lösungen zu extrahieren und auf den vorliegenden Fall anzuwenden.

Diese Arbeit gliedert sich wie folgt: In Kapitel 2 werden die Konzepte des BPMS, des CBR und des SCEM eingeführt und erläutert. Kapitel 3 zeigt auf, wie solch ein BPMS technisch mit einer CBR-Engine kombiniert werden kann, bevor in Kapitel 4 das 6R-Modell eines BPMS mit CBR-Engine eingeführt wird. Kapitel 5 fasst die Erkenntnisse zusammen und zeigt Ansatzpunkte für zukünftige Arbeiten zu diesem Thema auf.

10th International Conference on Wirtschaftsinformatik,
16th - 18th February 2011, Zurich, Switzerland

2. KONZEPTE

Relevant für die in Kapitel 3 und 4 vorgeschlagenen Modelle sind drei Konzepte: Business Process Management Systeme, (Supply Chain) Event Management und Case-based Reasoning. Das BPMS bildet die Basis für die Modellierung, Ausführung und Überwachung von Prozessen. Das SCEM zeigt Wege und Möglichkeiten auf, wie auf Ereignisse während der Ausführung von Prozessinstanzen reagiert werden kann. Beide zusammen liefern die Eingaben für das CBR, welches anhand der Prozessinstanz, des Prozessmodells und des eingetretenen Ereignisses nach vergleichbaren Fällen sucht und die gefundenen Lösungen auf den aktuellen Fall anwendet.

2.1 Business Process Management Systeme

Ein BPMS deckt im Gegensatz zu einem Workflowsystem den gesamten Bereich des Prozessmanagements in einem Unternehmen ab. Das heißt, das System erlaubt es dem Anwender, die unternehmerischen Prozesse in einer Prozesssprache zu modellieren, und ist gleichzeitig in der Lage, die modellierten Prozesse auszuführen, zu verwalten und zu überwachen [6]. Es sind daher verschiedene Komponenten in einem BPMS notwendig, die diese Aufgaben erfüllen (vergl. Abbildung 1).

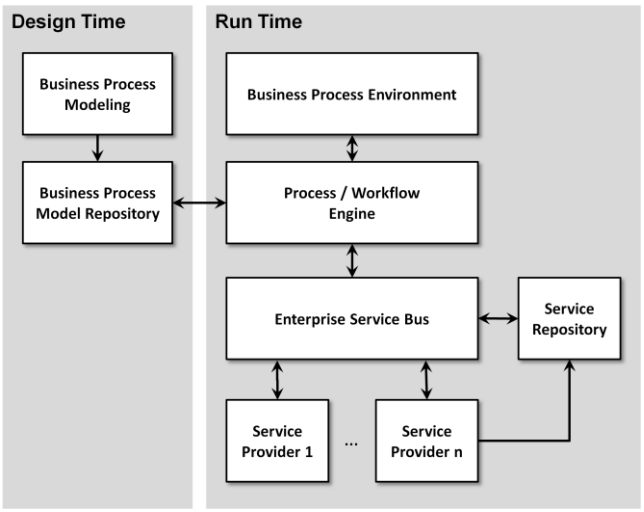

Abbildung 1: Modell eines BPMS kombiniert mit einer Serviceorientierten Architektur [6] [7]

Es gilt grundsätzlich zwei Bereiche zu unterscheiden: Den Bereich der Design Time und den Bereich der Run Time. Im Bereich der Design Time werden Prozesse durch die Anwender modelliert und anschließend in einem Repository gespeichert. Es wird also zu einem Zeitpunkt vor der Ausführung des Prozesses sein Design in Form eines Modells festgelegt. Im Bereich der Run Time gelangen die modellierten Prozesse dann zur Ausführung, wenn sie im Zuge der operativen Arbeit im Unternehmen benötigt werden. Damit deckt das System die Phasen Design & Analyse, Konfiguration und Ausführung aus dem Lebenszyklusmodell von Weske [6] ab. Die Überwachung und Bewertung der Prozesse kann anhand der gewonnenen Daten aus Prozessmodellierung und Prozessausführung als abschließende Phase im Lebenszyklus erfolgen.

Der Vorteil der Verwendung einer Serviceorientierten Architektur (SOA) in einem BPMS liegt in der damit verbundenen Flexibilität. Über den Enterprise Service Bus (ESB) steht eine standardisierte Kommunikationsschicht zur Verfügung. Services registrieren sich im Service Repository und werden über den Enterprise Service Bus aufgerufen, wenn sie benötigt werden [7]. Ändert sich etwas an einem Service oder wird ein Service durch einen neuen Service ersetzt, registriert sich der neue Service im Service Repository. Alle folgenden Aufrufe werden dann an den neuen Service weitergeleitet. An den Schnittstellen und den Geschäftsprozessen sind keine Anpassungen notwendig, wenn sich Services ändern. Es findet damit eine klare Trennung zwischen der Ablauflogik (den Unternehmensprozessen) und der Ablaufsteuerung (den Serviceaufrufen) statt. [8]

In vielen Fällen soll es in einem BPMS allerdings auch möglich sein, die Prozessinstanzen während der Laufzeit anzupassen, wenn etwa ein besonderes Ereignis aufgetreten ist, auf das im normalen Prozessablauf nicht reagiert werden kann. Lu et. al. schlagen dafür ein erweitertes BPMS-Konzept vor. Aus der Prozessausführung und der Prozessdiagnose heraus lassen sich Instanzen eines Prozesses adaptieren wenn dies notwendig wird. Diese Prozessadaptionen können dann auch Informationen für allgemeine Prozessverbesserungen liefern [5]. Werden etwa bei einem Prozess immer wieder dieselben Änderungen vorgenommen, dann empfiehlt sich in vielen Fällen eine Anpassung oder Erweiterung des zugrunde liegenden Prozessmodells.

2.2 Supply Chain Event Management

Unter dem Begriff des Supply Chain Event Managements wird die aktive Überwachung von Wertschöpfungsketten sowie das Management von auftretenden Störungen innerhalb dieser Ketten verstanden [9]. Es lassen sich dabei drei zentrale Funktionen des SCEM identifizieren [9]:

> **Monitoring**: Die zeitnahe Verwaltung, Visualisierung und Darstellung aller Transaktionen der Wertschöpfungskette, mit dem Ziel, die Überwachung und Verfolgung des gesamten Prozesses sicher zu stellen.
>
> **Track & Trace**: Die informationstechnische Abbildung des gesamten Prozesses von der ersten bis zur letzten Aktivität, sowie die regelmäßige Meldung des Prozessfortschritts.
>
> **Alert Management**: Wenn Störungen innerhalb des Prozesses auftreten bzw. es zu Abweichungen zwischen dem definierten Soll und dem Ist kommt, erfolgt eine Warnung an die verantwortlichen Prozessanwender.

Das SCEM bietet dabei eine Fülle an operativen Nutzungsmöglichkeiten im betrieblichen Umfeld, die von der Statusverfolgung von Kundenaufträgen über das Management kollaborativer, unternehmensübergreifender Prozesse bis hin zur Prozessdokumentation reichen [10]. Zwar bezieht sich das SCEM auf Wertschöpfungsketten in Unternehmen, diese können aber im breiteren Kontext als eine Spezialklasse von Unternehmensprozessen angesehen werden. Für den Anwendungsfall des flexiblen Prozessmanagements soll daher das Konzept des SCEM für Unternehmensprozesse allgemein angewendet werden.

Von besonderer Bedeutung ist das Alert Management, da es im Zuge der Ausführung von Prozessinstanzen dazu verwendet werden kann, um Abweichungen vom Soll des Prozesses anzuzeigen. Erst wenn durch das Alert Management eine Störung festgestellt wurde, kann mittels des CBR nach einer Lösung gesucht und diese auf den Prozess angewendet werden.

2.3 Case-based Reasoning

Bei CBR handelt es sich um eine Methode aus der kognitiven Forschung, mit der neue Probleme gelöst werden können, indem Lösungen von vergleichbaren Fällen aus der Vergangenheit auf das neue Problem angewendet werden [11]. Da es sich um eine Methode handelt, ist CBR nicht von einer spezifischen Technologie abhängig und kann in verschiedensten Systemen zur Anwendung kommen, wenn darin der CBR-Ablauf abgebildet werden kann [12].

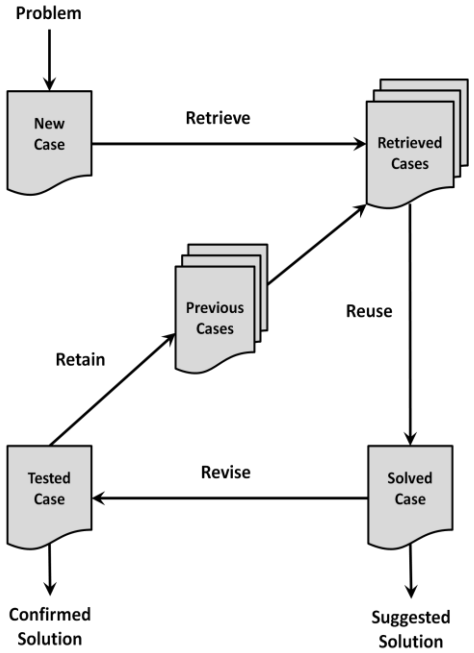

Abbildung 2: CBR-Prozess nach Aamodt und Plaza [13]

Ausgangspunkt für das CBR ist eine Falldatenbank, in der Fälle und ihre Lösungen gespeichert sind. Tritt nun ein neuer Fall auf, dann werden in der Falldatenbank Fälle mit vergleichbaren Eigenschaften gesucht. Es werden also Fälle aus der Vergangenheit ermittelt, die mit dem aktuellen Fall vergleichbar sind. Im zweiten Schritt wird dann versucht, die Lösung eines Falles aus der Vergangenheit auf das aktuelle Problem anzuwenden. Dazu muss die Lösung des vergangenen Falles eventuell noch an den neuen Fall angepasst werden. Nachdem die Lösung auf den neuen Fall angewendet worden ist, kann der Erfolg der Lösung evaluiert werden. Sollte sich die Lösung als erfolgreich herausstellen, wird der neue Fall samt seiner Lösung der Falldatenbank hinzugefügt und das System ist um einen Fall reicher. Durch den letzten Schritt baut das System mit jedem erfolgreich gelösten Fall seine Wissensdatenbank weiter aus und kann damit in der Zukunft noch treffsicherer neue Fälle erkennen und lösen [14].

Der zugehörige Prozess, aufgrund der vier Funktionen Retrieve, Reuse, Revise und Retain auch als 4R-Prozess bezeichnet, stellt sich dar wie in Abbildung 2 festgehalten [13].

Generell kann zu CBR festgehalten werden [15]:

> Ein Fall repräsentiert domänenspezifisches Wissen, welches an einen Kontext gebunden ist, und speichert dieses Wissen auf einer operationalen Ebene.
>
> Fälle können in unterschiedlicher Form und Größe auftreten und verschieden große Zeitspannen umfassen. Sie verbinden dabei Lösungen mit Problemen, Ergebnisse mit Situationen oder beides.
>
> Ein Fall speichert Erfahrungen, die von der Erwartung abweichen. Fälle, die es wert sind gespeichert zu werden, enthalten eine wertvolle Lektion.
>
> Wertvolle Lektionen sind solche Lektionen, die einem Entscheider helfen, in der Zukunft ein oder mehrere Ziele zu erreichen, ihn vor Fehlschlägen bewahren oder auf unvorhergesehene Probleme aufmerksam machen.

Für die Anwendung in einem BPMS ist der Eintritt eines Ereignisses in der Instanz eines Geschäftsprozesses der Ausgangspunkt für die Ermittlung von ähnlichen Fällen. Es geht darum, basierend auf dem aktuellen Ausführungszustand der Prozessinstanz Lösungsmöglichkeiten zu ermitteln, die für denselben oder einen ähnlichen Prozess in einem vergleichbaren Ausführungszustand und für ein ähnliches Ereignis in der Vergangenheit bereits erfolgreich waren.

Minor et. al. [16] beschreiben den Prozess dazu folgendermaßen:

„When a current workflow has to be adapted, it can be used as a query to the case base. The best matching cases are retrieved from the case base in a ranking induced by the similarity between the query and the problem part of the particular cases. The solution part of a case, that is, the adapted workflow revision, can be reused in order to change the query workflow." [16]

Um feststellen zu können, wie ähnlich sich zwei Fälle F und F' sind, müssen die Attribute f_i und f'_i paarweise miteinander verglichen werden. Dies kann über eine Ähnlichkeits- oder Distanzfunktion erreicht werden. Es wird dabei die Distanz zwischen den Attributen des aktuellen Falles F und eines Falles F' aus der Falldatenbank paarweise ermittelt und die Summe über die Distanzen aller Attributpaare gebildet. Um verschiedenen Attributen eine unterschiedliche Gewichtung zu geben, kann die Distanz zwischen zwei Attributen noch mit einer Gewichtung w multipliziert werden. Je ähnlicher sich zwei Fälle sind, umso kleiner ist die Distanz aller Attributpaare. Allgemein kann dies formuliert werden als

$$Sim(F, F') = \sum_{i=1}^{n} f(F_i, F'_i) \times w_i$$

wobei $f(F_i, F'_i)$ eine beliebige Distanzfunktion für das Attributpaar i der Fälle Z und F mit der Gewichtung w_i ist. Bei irrelevanten Attributen kann die Gewichtung w_i auf 0 gesetzt werden, damit sie nicht in die Bewertung der Ähnlichkeit einfließen [12].

3. TECHNISCHE ARCHITEKTUR

In Kapitel 2.1 wurde erläutert, wie die Architektur eines BPMS aufgebaut werden kann (vergl. Abbildung 1). Diese Architektur soll nun aufgegriffen und adaptiert werden. Sie besteht im Wesentlichen aus zwei großen Bereichen: einem Designmodul (Design Time) und einem Laufzeitmodul (Run Time). Das Designmodul dient dem Entwurf von Prozessmodellen und deren Speicherung. Das Laufzeitmodul lädt die Prozessmodelle aus dem Repository des Designmoduls und führt diese aus, indem es je nach Prozessbeschreibung verschiedene Services über den Enterprise Service Bus aufruft. Alle verfügbaren Services sind in einem Service Repository verzeichnet. Wird ein Service durch

einen anderen ersetzt, dann wird das Service Repository entsprechend aktualisiert, die Prozessmodelle selbst müssen nicht angepasst werden. [6] [7]

Die dargestellte BPMS-Architektur verfügt allerdings über keine Möglichkeit, Prozesse während der Laufzeit zu adaptieren bzw. flexibel auf Ereignisse im Sinne eines Event Managements zu reagieren. Dafür müssen im BPMS entsprechende Funktionen zur Erkennung von Störungen und zur Anpassung von Prozessinstanzen vorgesehen und die Software um ein CBR-Modul erweitert werden. Tritt im Ablauf einer Prozessinstanz ein besonderes Ereignis auf, soll das Business Process Environment die Möglichkeit haben, in der Falldatenbank nach ähnlichen Vorkommnissen zu suchen und die damals verwendeten Aktionen auf das aktuelle Ereignis anwenden. Es sind daher einige Erweiterungen an der Architektur des BPMS notwendig (vergl. Abbildung 3).

3.1 Event Listener

Adaptionen einer Prozessinstanz werden dann notwendig, wenn in einem Prozess ein Ereignis eintritt, welches im definierten Prozessablauf nicht behandelt werden kann. Es wird daher ein Event Listener benötigt, der laufend alle eintreffenden Ereignisse überwacht und überprüft, welche Prozessinstanzen von diesem Ereignis betroffen sind. Ereignisse können dabei direkt aus einer Prozessinstanz heraus entstehen, wenn zum Beispiel eine Laufzeitüberwachung ausgelöst wird, weil die Ausführung einer Aktivität zu lange dauert. Ebenso kann ein Ereignis aber auch von außerhalb des BPMS kommen, wenn etwa eine Ressource einen Defekt meldet oder der Kunde neue Daten zum Auftrag übermittelt. Der Event Listener informiert bei Eintritt eines Ereignisses das Business Process Environment über das Ereignis und welche Prozessinstanzen davon betroffen sind. Das Business Process Environment entscheidet dann, was aufgrund dieses Ereignisses zu tun ist.

3.2 Case Engine

Kann ein Ereignis nicht im Zuge der normalen Prozessausführung behandelt werden, übergibt das Business Process Environment eine Beschreibung der Prozessinstanz und deren Ausführungszustand an die Case Engine des CBR-Moduls. Die Case Engine ermittelt aus der Falldatenbank, dem sogenannten Case Repository, ähnliche Fälle aus der Vergangenheit und deren Lösung. Die Lösungen werden dem Business Process Environment von der Case Engine zur Verfügung gestellt. Das Business Process Environment kann dann entweder die am besten passende Lösung automatisch auf die Prozessinstanz anwenden oder mehrere Lösungen einem Anwender zur Entscheidung vorlegen. Konnte das aktuelle Ereignis durch die vorgeschlagene Lösung behoben werden, wird eine Beschreibung der Prozessinstanz, deren Ausführungszustand, eine Beschreibung des Ereignisses und der verwendeten Lösung im Case Repository gespeichert.

3.3 Case Repository

Eine Datenbank aller Fälle, die von der Case Engine verwaltet werden. Die Fallbeschreibung besteht aus einem Prozessmodell, einer Zustandsbeschreibung des Prozessmodells, einer Beschreibung des auslösenden Ereignisses und der verwendeten Lösung.

Zudem ist eine funktionale Erweiterung des Business Process Environment bzw. der Process Engine notwendig. Damit auf ein Ereignis flexibel reagiert werden kann, muss das Business Process Environment in der Lage sein, in Ausführung befindliche Prozessinstanzen zur Laufzeit zu adaptieren. Zum Beispiel sollen zusätzliche Aktivitäten zu einer Prozessinstanz hinzugefügt oder gelöscht werden. Erst wenn diese Voraussetzungen geschaffen sind, kann die neue Architektur funktionieren.

Kombiniert man nun die oben beschriebene Architektur mit dem Event Listener, der Case Engine und dem Case Repository und erweitern die Funktionalität des Business Process Environment und der Process Engine um die Möglichkeiten der Prozessadaptierung, dann erhält man eine Architektur, wie sie in Abbildung 3 dargestellt ist.

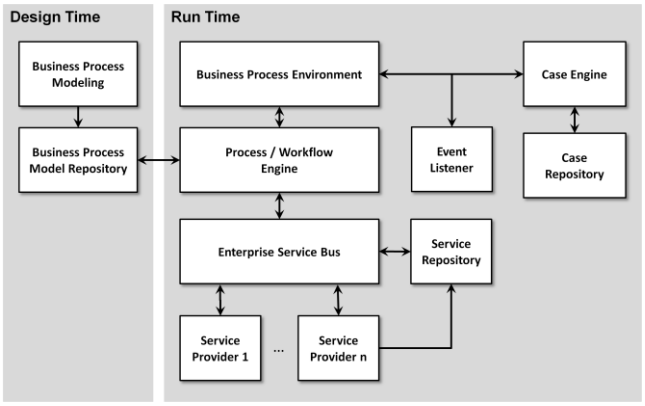

Abbildung 3: Architektur eines BPMS mit Ereignissteuerung und CBR

Die neuen Module müssen direkt an das Business Process Environment angeschlossen sein, da sie ihm die Möglichkeit geben, während der Ausführung einer Prozessinstanz dynamisch auf Events zu reagieren, indem die Prozessinstanz adaptiert wird. Etwa durch das Einfügen zusätzlicher Arbeitsschritte, durch das Adaptieren bestehender Arbeitsschritte, durch das Löschen von Arbeitsschritten oder den Austausch des Prozessmodells als Ganzes. Diese Eingriffe sind in verschiedenen Phasen der Prozessausführung möglich und eröffnen damit breitgefächerte Möglichkeiten zur Dynamisierung des Prozessmanagements.

3.4 Exception Handling

Ausgangspunkt für jede Fallbehandlung ist das Business Process Environment, da es die Ausführung der einzelnen Prozessinstanzen durch die Process Engine steuert. Wenn vom Event Listener ein Ereignis empfangen wird, dann muss das Business Process Environment entscheiden, wie es mit dem Ereignis umgeht. Dies hängt zunächst einmal primär von der Art des Ereignisses ab. Es lassen sich dabei fünf Kategorien von Ereignissen unterscheiden [17]:

Work Item Failure: Ein Work Item, also eine Aktivität im Prozess, kann nicht mehr weiter ausgeführt werden. Dies kann sich zum Beispiel in einem Abbruch durch den Anwender oder einem Hard- oder Softwarefehler äußern.

Deadline Expiry: Für eine oder mehrere Aktivitäten im Prozess kann ein Zeitlimit definiert werden. Das Zeitlimit gibt an, bis wann die Aktivität abgeschlossen sein sollte. Wird das Zeitlimit vor Abschluss der Aktivität erreicht, wird ein Ereignis ausgelöst.

Resource Unavailability: Aktivitäten sind in der Regel Ressourcen zugeordnet, die für die Ausführung der Aktivität verantwortlich sind. Dies können sowohl Anwender als auch technische Systeme sein. Ein Problem kann auftreten, wenn bei

der Zuordnung von Aktivitäten zu Ressourcen keine passende Ressource zur Verfügung steht oder wenn die Ressource während der Prozessausführung ausfällt.

External Trigger: Andere Prozesse, externe Systeme und Anwender können ebenfalls den Eintritt eines Ereignisses signalisieren. Speziell diese Ereignisse lassen sich schwer während des Entwurfs eines Prozessmodells vorhersehen und spielen daher eine wesentliche Rolle bei der flexiblen Behandlung von Prozessinstanzen.

Constraint Violation: Bei Constraints handelt es sich um Randbedingungen des Prozesses im Bezug auf die Aktivitäten, den Datenfluss oder die verknüpften Ressourcen, die eingehalten werden müssen um den konsistenten Ablauf und die Integrität des Prozesses sicherzustellen. Dies können etwa Regeln sein, die definieren in welcher Reihenfolge bestimmte Aktivitäten des Prozesses ausgeführt werden müssen und bei Verletzung dieser Regel ein Ereignis auslösen.

Tritt nun ein Ereignis aus einer dieser fünf Kategorien auf, dann gibt es im Rahmen des Supply Chain Event Management verschiedene Möglichkeiten, darauf zu reagieren [18]:

Repair: Eine Störung im Prozess wird durch einen direkten Eingriff in den aktuellen Arbeitsschritt der Prozessinstanz gelöst.

Reschedule: Die noch nicht ausgeführten Arbeitsschritte der Prozessinstanz werden neu geplant.

Replan: Die gesamte Prozessinstanz wird neu geplant, wenn ein Repair oder Reschedule nicht möglich ist. Es können zum Beispiel Teile der Prozessinstanz, die noch nicht ausgeführt wurden, durch andere Arbeitsschritte ersetzt werden.

Learn: Ist eine Korrektur der Prozessinstanz nicht möglich, dann können aus der Ausführung zumindest Lehren für die Zukunft gewonnen werden. Diese können zu einem späteren Zeitpunkt zum Beispiel als Basis für eine Prozessoptimierung oder ein Prozess Reengineering herangezogen werden.

Während es beim Repair und beim Reschedule nur zu Eingriffen in die Eigenschaften einzelner Prozessschritte oder eines Teils der Prozesskette kommt, wird bei Replan die gesamte Prozessinstanz angepasst. Bei der vierten Eingriffsmöglichkeit, dem Learn, geht es nicht mehr um die Anpassung einer einzelnen Prozessinstanz sondern um ein generelles Lernen aus Ereignissen. Treten zum Beispiel immer wieder dieselben Ereignisse an derselben Stelle in einem Prozess auf, dann können daraus Rückschlüsse auf das zugrunde liegende Prozessmodell gemacht werden. Dies kann zu einer Änderung oder einem Austausch des Prozessmodells führen, damit in Zukunft besser auf das Ereignis reagiert werden kann oder das Ereignis ganz vermieden wird.

Für den Eingriff in eine Prozessinstanz zum Zweck der Störungsbehebung stehen verschiedene Adaptionsmuster zur Verfügung, die von Weber et. al. vorgeschlagen wurden [19]. Diese Adaptionsmuster beschreiben, wie ein Prozess bei Bedarf adaptiert werden kann. Für eine umfängliche Anpassbarkeit der Prozesse im BPMS werden allerdings weitere Adaptionsmöglichkeiten benötigt. Zum einen muss auf der Ebene der Prozessaktivität eine Eingriffsmöglichkeit bestehen. Es soll möglich sein, dass die Daten der Aktivität und die Ressourcen angepasst werden können oder die Eigenschaften der Aktivität adaptierbar sind. Die Aktivitäten müssen zudem neu gestartet, abgebrochen oder abgeschlossen werden können. Zum anderen soll es auf Prozessebene möglich sein, im Falle einer Neuplanung des gesamten Prozesses das Prozessmodell gegen ein anderes auszutauschen. Es ergibt sich damit eine Liste an möglichen Eingriffsoptionen, wie sie in Tabelle 1 dargestellt sind.

Tabelle 1: Möglichkeiten der Adaption eines Prozesses

	Aktivität	Teil-prozess	Prozess
Aktivität reparieren			
Daten hinzufügen / ändern / löschen	✓		
Ressourcen hinzufügen / ändern / löschen	✓		
Aktivität neu starten	✓		
Aktivität abbrechen	✓		
Aktivität abschließen	✓		
Prozess anpassen			
Prozessfragment einfügen		✓	
Prozessfragment löschen		✓	
Prozessfragment verschieben		✓	
Prozessfragment ersetzen		✓	
Prozessfragment tauschen		✓	
Subprozess extrahieren		✓	
Subprozess einfügen		✓	
Prozessfragment in Schleife einbetten		✓	
Prozessfragment parallelisieren		✓	
Prozessfragment in Bedingung einbetten		✓	
Abhängigkeit hinzufügen		✓	
Abhängigkeit entfernen		✓	
Bedingung anpassen		✓	
Prozessmodell anpassen			
Prozessmodell austauschen			✓

Damit sind nun die möglichen Arten von Ereignissen bekannt [17] sowie die Möglichkeiten, wie auf diese Ereignisse reagiert werden kann [18]. In diesem Kapitel wurde bereits gezeigt, welche Erweiterungen an der BPMS-Architektur notwendig sind, um sie um ein CBR-Modul zu erweitern. Kombiniert man nun die Architektur mit der Methode des CBR, den möglichen Ereignissen und den Reaktionsmöglichkeiten, dann erhält man ein Modell für die Interaktion des BPMS mit dem CBR-Modul. Im folgenden Kapitel wird dargestellt, wie dieses Modell im Detail aussieht.

4. 6R-MODELL

Das Modell der Interaktion zwischen dem BPMS, speziell des Business Process Environment, und der CBR-Engine, orientiert sich am 4R-Modell des CBR, welches bereits in Kapitel 2.3 erläutert wurde. Das Modell kann zu einem 6R-Modell erweitert werden, um den spezifischen Anforderungen im Kontext des Prozessmanagements zu genügen und die Interaktionen zwischen dem BPMS und der CBR-Engine zu beschreiben.

Die Interaktion gliedert sich in sechs dezidierte Schritte, die das 6R-Modell ausmachen:

Receive Exception: Das Business Process Environment wird vom Event Listener über das Auftreten eines Ereignisses informiert.

Das Business Process Environment entscheidet, ob das Ereignis im Rahmen des definierten Prozessablaufs der betroffenen Prozessinstanz erledigt werden kann. Ist dies nicht der Fall, dann wird eine Beschreibung der Prozessinstanz, deren Ausführungszustand und eine Beschreibung des Ereignisses an die Case Engine übermittelt.

Retrieve Case: Anhand der Beschreibung der Prozessinstanz, deren Ausführungszustands und der Beschreibung des Ereignisses ermittelt die Case Engine ähnliche Fälle aus der Vergangenheit aus dem Case Repository. Wie dies im Detail funktioniert, wird in Kapitel 4.1 eingehend erläutert. Die Fälle aus dem Case Repository enthalten jeweils eine Beschreibung der Lösung, die in der Vergangenheit angewendet wurde. Es handelt sich dabei um eine oder mehrere der Adaptionsmöglichkeiten der Prozessinstanz (vergl. Tabelle 1).

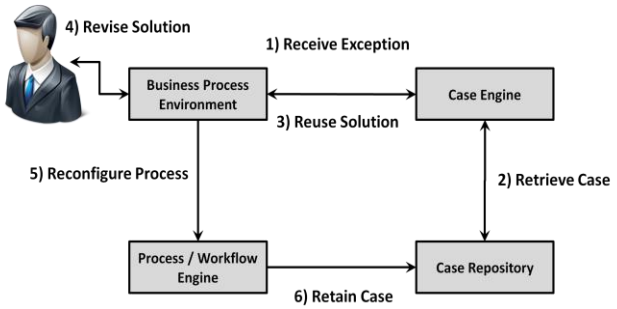

Abbildung 4: Interaktion zwischen dem BPMS und der Case Engine

Reuse Solution: Die möglichen Lösungen, die aus dem Case Repositiory ermittelt wurden, werden von der Case Engine an das Business Process Environment übergeben. Das Business Process Environment entscheidet nun anhand definierter Regeln, ob die am besten passende Lösung automatisch auf die Prozessinstanz angewendet wird oder ob ein Anwender entscheiden soll, welche Lösung verwendet wird. Ist eine Überprüfung durch den Anwender notwendig, wird der Schritt „Revise Solution" ausgeführt, andernfalls wird mit „Reconfigure Process" fortgefahren.

Revise Solution: Ein Anwender wird über das Vorliegen eines Ereignisses informiert. Gleichzeitig präsentiert ihm das BPMS die Lösungsvorschläge, die aus dem Case Repository ermittelt wurden. Der Anwender kann nun entscheiden, welche Lösung auf den vorliegenden Fall angewendet werden soll. Ist der Anwender mit keiner der vorgeschlagenen Lösungen einverstanden, kann er entweder eine neuerliche Ermittlung ähnlicher Fälle aus der Vergangenheit durch die Case Engine veranlassen oder selbst eine Lösung erfassen.

Reconfigure Process: Die ermittelte Lösung wird auf den aktuellen Fall, die betroffene Prozessinstanz, angewendet. Die Anpassung kann von einer Änderung an der aktuellen Aktivität bis hin zu einem Neustart des Prozesses mit einem anderen Prozessmodell reichen. Die vorgenommenen Anpassungen sind eine Kombination aus den Anpassungsmöglichkeiten, die bei Prozessschritt 2 angeführt wurden.

Retain Case: Wenn die Lösung erfolgreich auf die Prozessinstanz angewendet werden konnte, dann wird eine Beschreibung der Prozessinstanz, deren Zustand zum Zeitpunkt des Ereigniseintritts und die Lösung an das Case Repository übergeben und abgespeichert. Wenn in Zukunft ein ähnlicher Fall auftritt, kann die Lösung des aktuellen Falls dort wieder verwendet werden.

Von besonderer Bedeutung sind in diesem Ablauf zwei Punkte: Die Ermittlung ähnlicher Fälle aus dem Case Repository und die Adaption der Prozessinstanz.

4.1 Ähnlichkeitsfunktion

Der Vergleich zweier Fälle erfolgt immer über deren Charakterisierung. Es wird dazu über eine Ähnlichkeitsfunktion verglichen, inwieweit die Charakterisierungsteile der beiden Fälle übereinstimmen. Die allgemeine Form einer Ähnlichkeitsfunktion wurde bereits in Kapitel 3.4 erläutert und sieht folgendermaßen aus:

$$Sim\ (C, C') = \sum_{i=1}^{n} f(p_i, p'_i) \times w_i$$

Es werden dabei paarweise die einzelnen Attribute der Charakterisierungsteile C und C' verglichen, um die Ähnlichkeit der beiden Fälle ermitteln zu können. Je ähnlicher sich der Fall aus der Falldatenbank und der aktuelle Fall sind, umso besser wird das Ergebnis der Ähnlichkeitsfunktion $Sim(C, C')$ sein.

Für den Fall einer Prozessinstanz in der ein Ereignis oder eine Störung auftrat, ist die Beschreibung des Charakterisierungsteils nicht mehr so einfach möglich. Die Charakterisierung des aktuellen Falls muss über eine Beschreibung der Prozessinstanz, deren Ausführungszustand und eine Beschreibung des Ereignisses mit Fällen aus der Falldatenbank des CBR verglichen werden.

Wie dies in der Praxis funktionieren kann, soll hier exemplarisch anhand einer Prozessbeschreibung mittels Petri-Netzen illustriert werden:

Definition 1: WN ist ein Petri-Netz mit (P, T, F), für das gilt:

P ist die Menge der Plätze. Ein Platz stellt einen Status des Prozesses dar.

T ist die Menge der Transitionen. Eine Transition stellt einen Arbeitsschritt des Prozesses dar.

F ist die Menge der Pfeile zwischen Plätzen und Transitionen, wobei $F \subseteq (P \times T) \cup (T \times P)$. Die Pfeile geben die Abfolge der Stati und Arbeitsschritte des Prozesses vor.

WN verfügt über einen Quellplatz i und einen Senkenplatz o, mit $\bullet i = \emptyset$ und $o\bullet = \emptyset$.

Wenn eine Transition t^* zu WN hinzugefügt wird, die die Plätze i und o verbindet (mit $\bullet t^* = \{o\}$ und $t^*\bullet = \{i\}$), dann ist das resultierende Petri-Netz stark verknüpft.

Für alle Plätze $p \in P$ gilt, $M(p) \leq 1$. Das Abfeuern einer Transition im Zustand M führt zu einem Zustand M', bei dem $(\forall p \in \bullet t)\ M'(p) = M(p) - 1 \wedge (\forall p \in t\bullet)\ M'(p) = M(p) + 1$

Zusätzlich wird der Ausführungszustand des Prozesses und eine Beschreibung des aufgetretenen Ereignisses benötigt. Daraus und aus dem Prozessmodell in Form eines Petri-Netzes lässt sich der Charakterisierungsteil des Falles konstruieren:

Definition 2: Der Charakterisierungsteil eines Falles ist ein Tupel $C = (ID, WN, M, E)$, für das gilt:

ID ist die eindeutige Identifikation des Prozessmodells aus dem Business Process Model Repository, auf dem die Prozessinstanz basiert

$WN = (P, T, F)$ ist das CEW-Netz der Prozessinstanz mit den Plätzen P, den Transitionen T und den Pfeilen F

$M = (m_1, m_2, \dots, m_n)$ ist der aktuelle Zustand des CEW-Netzes, mit $M(p_i) = m_i$ als Zustand des Platzes p_i

$E = (e_1, e_2, \dots, e_m)$ ist das aufgetretene Event, welches eine Anpassung oder Änderung der Prozessinstanz notwendig macht, mit den Eigenschaften e_1 bis e_m

Der Charakterisierungsteil besteht damit aus drei unterschiedlichen Teilen, die im Rahmen des CBR unterschiedlich behandelt werden müssen. Es bietet sich daher an, die drei Bereiche des Charakterisierungsteils getrennt mit den entsprechenden Teilen der Charakterisierung von Fällen aus der Falldatenbank zu vergleichen. Damit ergibt sich für die Ähnlichkeitsfunktion folgende, etwas umfangreichere Form:

$$Sim(C, C') = f(WN, WN') + g(M, M') + \sum_{j=1}^{m} h(e_j, e'_j) \times w_j$$

Die Funktionen f, g und h können je nach Bedarf und Anforderung individuell gestaltet werden. Exemplarisch soll hier ein einfaches Beispiel für die Gestaltung dieser Funktionen erläutert werden, je nach Anwendungsfall muss aber eine spezielle Ähnlichkeitsfunktion entwickelt werden, die an die spezifischen Bedürfnisse des jeweiligen Unternehmens angelehnt ist.

Funktion f (Vergleich der Prozessmodelle): Über die Funktion f werden die beiden Petri-Netze miteinander verglichen. Hierfür können besonders einfach Mengenoperationen verwendet werden. Dazu wird die Funktion f in drei Teilfunktionen zerlegt:

$$f(WN, WN') = f_1(P, P') + f_2(T, T') + f_3(F, F')$$

wobei die drei Teilfunktionen jeweils die symmetrische Differenz der Menge der Plätze (P und P'), der Transitionen (T und T') und der Pfeile (F und F') bilden und die Mächtigkeit der entstandenen Mengen, also die Anzahl der verbleibenden Elemente, ermitteln. Sind z.B. die Plätze in den Mengen P und P' identisch, dann ist die symmetrische Differenz der beiden Mengen leer und damit die höchstmögliche Ähnlichkeit gegeben (die symmetrische Differenz hat in diesem Fall die Mächtigkeit 0). Die Funktion f_1 liefert für diesen Fall als Ergebnis also 0. Enthält die symmetrische Differenz dagegen Elemente, dann liefert die Funktion abhängig von der Anzahl der enthaltenen Elemente ein Ergebnis größer 0.

Die Teilfunktionen lassen sich daher charakterisieren als:

$$f_1(P, P') = |P \Delta P'|$$
$$f_2(T, T') = |T \Delta T'|$$
$$f_3(F, F') = |F \Delta F'|$$

Damit ergibt sich für die Funktion f:

$$f(WN, WN') = |P \Delta P'| + |T \Delta T'| + |F \Delta F'|$$

Liefert die Funktion f als Ergebnis 0, dann sind die beiden verglichenen Netze WN und WN' identisch, das heißt, sie enthalten dieselben Plätze, Transitionen und Pfeile.

Funktion g (Vergleich der Ausführungszustände): Über die Funktion g wird ermittelt, wie weit die Zustände M und M' der beiden Prozesse voneinander abweichen. Dabei werden die Funktionen M und M' nur auf jene Plätze P^* angewendet, für die gilt $P^* = P \cap P'$. Es werden im Vergleich also nur jene Plätze berücksichtigt, die in beiden Netzen vorkommen. Für den Fall, dass $M = M'$ ist, soll die Funktion als Ergebnis 0 liefern. Je weiter die Zustände voneinander abweichen, desto größer soll das Ergebnis der Funktion sein. Die einfachste Form der Funktion g wäre ein paarweiser Vergleich von m_j und m'_j mit

$$g(M, M') = \sum_{i=1}^{n} g'(m_i, m'_i)$$

wobei $g'(m_i, m'_i) = 0$, wenn $m_i = m'_i$ und $g'(m_i, m'_i) = 1$, wenn $m_i \neq m'_i$ ist. Die Verwendung einer Gewichtung macht in diesem Fall keinen Sinn, da die Zustände aller Plätze p und p' gleichwertig sind und damit auch die Ergebnisse für $g'(m_i, m'_i)$.

Durch eine aufwändigere Funktion g ließe sich nicht nur ermitteln, ob M und M' voneinander abweichen, sondern auch, inwieweit sich der Ausführungsgrad der beiden Prozesse unterscheidet. Dies lässt sich aber nicht mehr ohne weiteres mathematisch formulieren und muss durch einen Algorithmus in der Case Engine gelöst werden. Die Gestaltung dieses Algorithmus kann dabei individuell nach den Anforderungen des jeweiligen Unternehmens erfolgen.

Ebenso ist ein aufwändigerer Vergleich des Ausführungszustands der beiden Prozessinstanzen notwendig, wenn die Petri-Netze WN und WN' wesentlich voneinander abweichen und in den Vergleich der Zustände auch die Prozessmodelle selbst eingebunden werden müssen.

Funktion h (Vergleich der Ereignisse): Über die Funktion h wird schlussendlich erhoben, inwiefern sich die beiden auslösenden Ereignisse voneinander unterscheiden. Die Definition einer spezifischen Funktionen ist hier schwierig, da die Gestaltung der Funktion von den definierten Eigenschaften e_j und e'_j und der Art des Ereignisse E und E' abhängt. Für den Vergleich der Ereignisse soll daher an dieser Stelle die generische Form beibehalten werden, da davon auszugehen ist, dass nur solche Fälle verglichen werden, in denen dasselbe Ereignis aufgetreten ist und die damit auch über dieselben Ereigniseigenschaften verfügen:

$$Sim(E, E') = \sum_{j=1}^{m} h(e_j, e'_j) \times w_j$$

Damit lässt sich die gesamte Ähnlichkeitsfunktion als Summe der Funktionen f, g und h darstellen:

$$Sim\,(C, C') = |P \Delta P'| + |T \Delta T'| + |F \Delta F'| + \sum_{i=1}^{n} g'(m_i, m'_i) + \sum_{j=1}^{m} h(e_j, e'_j) \times w_j$$

Ergibt die Ähnlichkeitsfunktion für zwei Fälle mit den Charakterisierungsteilen C und C' den Wert 0, dann sind die beiden Fälle identisch. Sowohl die beiden Prozessmodelle als auch deren Ausführungszustand und die Ereignisse sind identisch. Je größer jedoch der Unterschied ist, desto größer auch das Ergebnis der Ähnlichkeitsfunktion.

Die notwendige Form der Ähnlichkeitsfunktion muss im Rahmen der Einführung eines solchen Systems analysiert, definiert und implementiert werden. Da die Ähnlichkeitsfunktion regelt, welche Fälle als vergleichbar angesehen werden, stellt sie die zentrale Funktionalität einer Case Engine dar. Entspricht die Ähnlichkeitsfunktion nicht den Anforderungen des jeweiligen Unternehmens, dann wird sich dies in einer mangelhaften Qualität der Suchergebnisse der Case Engine niederschlagen. Es ist daher bei der Konzeption der Ähnlichkeitsfunktion besondere Aufmerksamkeit geboten.

4.2 Prozessadaptionen

Wenn über eine Ähnlichkeitsfunktion erst einmal vergleichbare Fälle aus der Vergangenheit ermittelt sind, können potentielle Lösungen aus diesen Fällen abgeleitet werden. Im Fall von Ereignissen innerhalb von Geschäftsprozessen muss im Zuge der Prozessausführung auf die Ereignisse reagiert werden. Dies ist entweder im Rahmen des definierten Prozessmodells bereits möglich oder erfordert eine Adaption der Prozessinstanz.

Tabelle 2: Übersicht der Eingriffsmöglichkeiten und zugehörigen Funktionen

	Funktion
Aktivität reparieren	

Daten hinzufügen / ändern / löschen	`insertDataset`
	`updateDataset`
	`deleteDataset`

Ressourcen hinzufügen / ändern / löschen	`insertResource`
	`updateResource`
	`deleteResource`

Aktivität neu starten	`restartActivity`
Aktivität abbrechen	`cancelActivity`
Aktivität abschließen	`finishActivity`
Prozess anpassen	
Prozessfragment einfügen	`insertActivity`
Prozessfragment löschen	`deleteActivity`
Prozessfragment verschieben	`deleteActivity` `insertActivity`
Prozessfragment ersetzen	`deleteActivity` `insertActivity`
Prozessfragment tauschen	`deleteActivity` `insertActivity`
Subprozess extrahieren	`deleteActivity` `insertActivity`
Subprozess einfügen	`deleteActivity` `insertActivity`
Prozessfragment in Schleife einbetten	`insertLoop`
Prozessfragment parallelisieren	`insertDependency` `deleteDependency`
Prozessfragment in Bedingung einbetten	`insertCondition`
Abhängigkeit hinzufügen	`insertDependency`
Abhängigkeit entfernen	`deleteDependency`
Bedingung anpassen	`updateCondition`
Prozessmodell anpassen	
Prozessmodell austauschen	`replaceProcessModel`

Es wurde bereits in 3.4 erläutert (vergl. Tabelle 1), welche Adaptionsmöglichkeiten auf Aktivitäten, Teilprozess- und Prozessebene vorzusehen sind. Diese müssen als Funktionen im BPMS zur Verfügung gestellt werden und es erlauben, Prozessinstanzen während deren Ausführung anzupassen. Dabei muss nicht für jedes Adaptionsmuster eine eigene Funktion implementiert werden. Vielmehr können viele Adaptionsmuster aus einigen, wenigen atomaren Funktionen zusammengestellt werden. Tabelle 2 stellt die Adaptionsmuster den notwendigen Funktionen gegenüber.

Die durchgeführten Adaptionen an der Prozessinstanz fließen als Lösung wieder in das Case Repository ein und stehen dann für neue Fälle zur Verfügung. Damit kann sichergestellt werden, dass erworbenes Wissen zur erfolgreichen Behandlung eines Ereignisses innerhalb eines Prozesses nicht verloren geht, sondern erhalten bleibt.

5. Zusammenfassung & Ausblick

Der Wunsch nach standardisierten Unternehmensprozessen bei gleichzeitiger Möglichkeit zur individuellen Fallbehandlung von Ereignissen über den Rahmen der definierten Prozesse hinaus bildete die einleitende Prämisse dieser Arbeit. CBR, implementiert in einem BPMS, erlaubt es, Prozesse nach fest definierten, standardisierten Prozessmodellen auszuführen und trotzdem individuell auf Ereignisse reagieren zu können. Solange keine besonderen Ereignisse in der Prozessausführung auftreten, führt das BPMS jede Prozessinstanz anhand der definierten Prozessmodelle aus. Erst bei Eintreten eines besonderen Ereignisses, auf das nicht im Zuge des definierten Prozessmodells in adäquater Weise reagiert werden kann, kommt die Case Engine zum Einsatz. Sie ermittelt ähnliche Fälle aus der Vergangenheit und versucht, die damals erfolgreichen Lösungen auf den aktuellen Fall anzuwenden. Dies bietet den Vorteil, dass zum Zeitpunkt der Prozessmodellierung nicht alle möglichen Ereignisse, die während der Prozessausführung auftreten können, antizipiert werden müssen. Vielmehr wird nur der bekannte Standardprozess definiert und auf Ereignisse fallbasiert reagiert. CBR erweist sich dadurch als sinnvolle Ergänzung eines BPMS, wenn der Bedarf nach ereignisgesteuerter Behandlung von Ausnahmen über den definierten Prozess hinaus besteht.

Die große Herausforderung innerhalb des CBR stellt die Ermittlung von ähnlichen Fällen aus dem Case Repository mittels einer Ähnlichkeitsfunktion dar. In Kapitel 4.1 wurde in vereinfachter Form aufgezeigt, wie dies mittels Petri-Netzen für einfache Prozessmodelle aussehen könnte. Wenn den einzelnen Transitionen und Plätzen eines Petri-Netzes Eigenschaften zugeordnet werden, die für eine Prozessbeschreibung notwendig sind, gestaltet sich die Ermittlung von ähnlichen Fällen deutlich aufwändiger. Weitere Arbeiten auf diesem Gebiet sollen Klarheit darüber bringen, wie Prozesse sinnvoll mit mathematischen, formalen Mitteln (z.B. in Form von Graphen) beschrieben werden können, damit sie als Basis für eine Ähnlichkeitsfunktion tauglich sind und gleichzeitig als probates Mittel zur Prozessmodellierung dienen können.

In einem zweiten Schritt wird dann zu untersuchen sein, wie solch formale Modelle auf etablierte Prozessmodellierungssprachen wie BPMN angewendet werden können, da diese die Basis vieler am Markt befindlicher BPMS darstellen. Ebenso gilt es zu klären, wie Datenformate zur Kommunikation zwischen BPMS und Case Engine aussehen können, in welcher Form Fälle im Case Repository sinnvollerweise gespeichert werden und wie bei der Adaption von Prozessinstanzen sichergestellt werden kann, dass die Prozessmodelle ausführbar sind.

6. Literaturverzeichnis

[1] van der Aalst, W. M., ter Hofstede, A., & Weske, M. 2003. Business Process Management: A Survey. *International Conference on Business Process Management* , S. 1-12.

[2] van der Aalst, W. M. 2004. Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Worklfow Management. In *J. Jorgensen, & C. Neumair, Lectures on Concurrency and Petri Nets*. Berlin, Heidelberg: Springer.

[3] van der Aalst, W. M. 2004. Business Process Management: A Personal View. *Business Process Management Journal* , Vol. 10.

[4] Weber, B., Rinderle, S., Wild, W., & Reichert, M. 2005. CCBR-Driven Business Process Evolution. *Proceedings of the 6th International Conference on Case-Based Reasoning*, S. 610-624.

[5] Lu, R., Sadiq, S., & Governatori, G. 2009. On Managing Business Process Variants. *Data & Knowledge Engineering* Vol. 68, No. 7 , S. 642-664.

[6] Weske, M. 2007. *Business Process Management: Concepts, Languages, Architectures*. Berlin, Heidelberg: Springer.

[7] vom Brocke, J. 2008. *Serviceorientierte Architekturen - SOA: Management und Controlling von Geschäftsprozessen*. München: Franz Vahlen.

[8] Leymann, F. 2003. Web Services: Distributed Applications without Limits. An Outline. *10th Conference on Database Systems for Business, Technology and Web*. Berlin.

[9] Hellingrath, B., Laakmann, F., & Nayabi, K. 2004. Auswahl und Einführung von SCM-Softwaresystemen. In *H. Beckmann, Supply Chain Management: Strategien und Entwicklungstendenzen in Spitzenunternehmen*. Berlin, Heidelberg: Springer.

[10] Nissen, V. 2002. Supply Chain Event Management. *Wirtschaftsinformatik*, Vol. 44 , S. 477-480.

[11] Watson, I., & Marir, F. 1994. Case-based Reasoning: A Review. *The Knowledge Engineering Review*, Vol. 9 No. 4 , S. 327-354.

[12] Watson, I. 1999. Case-based Reasoning Is a Methodology not a Technology. *Knowledge-Based Systems*, Vol. 12 , S. 303-308.

[13] Aamodt, A., & Plaza, E. 1994. Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, Vol. 7 , S. 39-59.

[14] Finnie, G., & Sun, Z. 2003. R5 Model for Case-Based Reasoning. *Knowledge-Based Systems*, Vol. 16 , S. 59-65.

[15] Kolodner, J. L. 1993. *Case-based Reasoning*. San Mateo: Morgan Kaufmann.

[16] Minor, M., Tartakovski, A., Schmalen, D., & Bergmann, R. 2008. Agile Workflow Technology and Case-Based Change Reuse for Long-Term Processes. *International Journal of Intelligent Information Technologies*, Vol. 4, No. 1, S. 80-98.

[17] Russell, N., van der Aalst, W. M., & ter Hofstede, A. H. 2006. Workflow Exception Patterns. *Proceedings of the 18th Conference on Advanced Information Systems Engineering* , S. 288-302.

[18] Otto, A. 2003. Supply Chain Event Management: Three Perspectives. *The International Journal of Logistics Management*, Vol. 14, No. 2 , S. 1-13.

[19] Weber, B., Rinderle, S., & Reichert, M. 2007. Change Patterns and Change Support Features in Process-Aware Information Systems. *Proceedings of the 19th Conference on Advanced Information Systems Engineering* , S. 574-588.