TECHNISCHE UNIVERSITÄT DRESDEN

Software Engineering Ubiquitärer Systeme

Großer Beleg

Betrachtung von MAPE-K Kontrollschleifen für Prozesse in CPS

Betreuer

Dipl.-Inf. Ronny Seiger,
Dipl.-Medieninf. Steffen Huber

Verantwortl. Hochschullehrer

Prof. Dr. Thomas Schlegel

1. Februar 2016

Peter Heisig

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel

Betrachtung von MAPE-K Kontrollschleifen für Prozesse in CPS

unter Angabe aller Zitate und nur unter Verwendung der angegebenen Literatur und Hilfsmittel selbstständig angefertigt habe.

Dresden, den 1. Februar 2016

Peter Heisig

01.06.2015

Betrachtung von MAPE-K Kontrollschleifen für Prozesse in CPS

Aufgabenstellung Großer Beleg

Name, Vorname: Heisig, Peter

Studiengang: Diplom Informatik PO 2004

Bearbeitungszeitraum: 01.06.2015 bis 30.11.2015
Betreuer/in: Dipl.-Inf. Ronny Seiger, Dipl.-Medieninf. Steffen Huber
Betreuender Hochschullehrer: Jun.-Prof. Dr.-Ing. Thomas Schlegel
Institut: Software- und Multimediatechnik

Hintergrund

MAPE-K Kontrollschleifen beschreiben Regelkreise für selbst-adaptive Systeme. Dabei werden die Phasen *Monitor*, *Analyze*, *Plan* und *Execute* iterativ für ein autonom arbeitendes, selbst-regulierendes System durchlaufen. Mit Cyber-physical Systems (CPS) geht die zunehmende Vernetzung von virtuellen Komponenten (Software) und realen Geräten und Gegenständen vertreten durch beobachtende Komponenten (Sensorik) und ausführende Komponenten (Aktuatorik) einher. Hiermit lassen sich MAPE-K Kontrollschleifen auch auf intelligente Umgebungen (Smart Homes, Smart Factories) bestehend aus virtuellen und realen Entitäten, die sich wechselseitig beeinflussen, anwenden.

Zur Automatisierung von sich wiederholenden Abläufe kommen Prozesse bzw. Workflows zum Einsatz. Im Web- bzw. SOA-Bereich werden Business Processes angewandt um die Ausführung von Web Services zu orchestrieren und so Geschäftsprozesse von Unternehmen abzubilden. Diese Prozesse und ihre Aktivitäten bestehen aus Aufrufen an virtuelle Server- bzw. Software-Komponenten, die der Daten- und Kontrollflussmanipulation dienen. Die Überwachung der Ausführung (*Monitoring*) von Prozessen kann daher rein durch die genutzten Services bzw. durch Software stattfinden. Bei der Nutzung von Prozessen zur Automatisierung von Abläufen kann es jedoch auch zur Manipulation von Real-Welt-Objekten kommen. Die Korrektheit der ausgeführten Aktivitäten mit Einfluss auf die physische Welt kann nicht ausschließlich durch Server-Antworten überprüft werden, weshalb zusätzliche Mechanismen zur Überprüfung der erfolgten Real-Welt-Manipulationen und damit der Konsistenzsicherstellung zwischen Cyber- und physischer Welt zum Einsatz kommen müssen.

Aufgabenstellung

Im Rahmen dieser Belegarbeit ist die Anwendung von MAPE-K Kontrollschleifen für Prozesse in Cyber-physical Systems zu betrachten. Dabei sollen die einzelnen Phasen bei der Prozessausführung identifiziert und Konzepte zur Integration der MAPE-K Phasen in CPS Prozesse entwickelt werden.

Hierzu sind zunächst grundlegende Recherchen zu den Themen CPS und Internet of Things, MAPE-K und Business Process Management durchzuführen und entsprechende Anforderungen für die Abbildung von MAPE-K Schleifen in CPS Prozesse aufzustellen. Anschließend sollen verwandte Forschungsarbeiten hinsichtlich der identifizierten Anforderungen untersucht und bewertet werden.

Im Hauptteil der Arbeit findet die genaue Untersuchung von MAPE-K Kontrollschleifen für CPS-Prozesse statt. Dabei sind die einzelnen Phasen zu identifizieren und Vorschläge zur Umsetzung dieser Phasen für Prozesse in CPS zu entwickeln. Dabei soll sowohl auf die Aspekte der Prozessmodellierung als auch auf die Prozessausführung eingegangen werden. Es ist zu überprüfen, inwieweit die vorhandene Sensorik und Aktuatorik genutzt werden kann um die MAPE-K Phasen umzusetzen und so eine konsistente Abbildung zwischen virtueller und physischer Welt bei der Prozessausführung – auf der Ebene von Aktivitäten, Unterprozessen und Prozessen – zu erreichen. Dabei sind auch transaktionale Eigenschaften von CPS Prozessen zu diskutieren. Weiterhin ist zu eruieren, ob Simulationsverfahren während der *Plan*-Phase eingesetzt werden können um Voraussagen über das Ergebnis der Ausführung von Prozessschritten zu treffen und folgend bzw. während der Ausführung eine eventuell notwendige Adaption durchzuführen.

Basierend auf den Erkenntnissen und Entwicklungen der Konzeptionsphase ist ein vertikaler Prototyp zur Prozessausführung in Cyber-physical Systems anhand eines selbstgewählten Beispiels aus dem Bereich Smart Home umzusetzen. Hierbei soll vorhandene Aktuatorik und Sensorik zur Illustration der MAPE-K Phasen bei der Ausführung von CPS Prozessen eingesetzt werden. Ein bereits existierendes Metamodell sowie eine Ausführungsumgebung für Prozesse dienen als Grundlage für die Prototypen. Das Metamodell und die Ausführungsumgebung sind entsprechend der Ergebnisse aus den vorangegangenen Arbeitsschritten zu erweitern.

In der Ausarbeitung zur Belegarbeit werden die Ergebnisse der Anforderungsanalyse und der Literaturrecherche, eine Übersicht zum Stand der Technik sowie die in dieser Arbeit erstellten Konzepte und deren Implementierung beschrieben. Aufgetretene Probleme während der Realisierung und die umgesetzten Lösungen werden ebenfalls dargestellt und diskutiert. Eine Evaluation und Diskussion des Konzeptes anhand der aufgestellten Anforderungen sowie der Umsetzung im Prototyp schließen die Ausarbeitung gemeinsam mit einem Ausblick auf weiterführende Arbeiten ab.

Jun.-Prof. Dr. -Ing. Thomas Schlegel	Peter Heisig

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Intelligente Umgebungen, wie *Smart Home (SH)* oder *Smart Factories*, sind seit längerer Zeit im produktiven Einsatz. Die Automatisierung ihrer Methoden und Verfahren wird aufsetzend auf der IT-Infrastruktur immer öfter durch Prozesse abgebildet, um Kosten zu senken und Zeit zu sparen. Unter der Nutzung einer Vielzahl von Ressourcen und externer Services ermöglichen sie Arbeitsabläufe in heterogenen Netzen von Systemen. Die zunehmende Komplexität solcher verteilter Systeme erfordert ein hohes Maß an Kontrolle und die Reflexion ausgeführter Aktionen. Cyber-physische Systeme (CPS) als Teil der Lösung der damit einhergehenden Probleme und Bindeglied zwischen Realität und der virtuellen Welt, besitzen die Fähigkeit Objekte und deren Zustände mittels Aktuatoren zu manipulieren. Da diese jedoch nicht imstande sind ihr Wirken zu überprüfen, müssen Sensoren in eine Rückkopplung bezüglich ausgeführter Operationen eingebunden werden. Um die Abbildung der Realität innerhalb des Systems konsistent zu gewährleisten, können Regelkreise in den Ablauf von Aktionen integriert werden. Diese Fähigkeit der Selbstregulation von Prozessen innerhalb von CPS gewinnt mit deren vermehrtem Einsatz an Relevanz und bedarf einer expliziten Untersuchung.

1.2 Zielsetzung

Für die Integration der Phasen Monitor, Analyze, Plan, Execute und Knowledge (MAPE-K) in CPS-Prozesse werden Anforderungen erhoben und verwandte Forschungsarbeiten im Hinblick auf deren Eignung untersucht. Die Identifikation und konzeptionelle Integration der Phasen in ein bestehendes Prozessmodell beinhaltet unter anderem die Aspekte der Modellierung, als auch die der Ausführung von Prozessen. Bereits vorhandene Sensorik und Aktuatorik wird bei der Integration in das selbstadaptive System berücksichtigt. Weiterhin sind transaktionale Eigenschaften der Prozesse in CPS von Bedeutung, wie auch die Möglichkeiten der Simulation von Laufzeitadaptionen in der Planungsphase von MAPE-K. Es wird ein

existierendes Prozessmodell, sowie dessen Ausführungsumgebung um die Integration der Kontrollschleife erweitert. Dieser vertikale Prototyp wird anhand der erhobenen Anforderungen, wie auch im Bezug auf Probleme der Umsetzung diskutiert.

1.3 Aufbau der Arbeit

In dieser Arbeit werden zunächst die Themen CPS und das Internet of Things (IoT), sowie Business Process Management (BPM) und MAPE-K Kontrollschleifen umrissen. Die in Kapitel 2 diskutierten Grundlagen, bildet in Kombination mit dem aktuellen Forschungsstand aus Abschnitt 3 die Basis für Anforderungen an Konzept und Technologie. Die beleuchteten wissenschaftlichen Arbeiten werden auf ihre Eignung und den konzeptuellen Einsatz überprüft. Unter Verwendung der Ergebnisse vorangegangener Untersuchungen, wird die Integration der Kontrollschleife in CPS-Prozesse detailliert erläutert. In Kapitel 6 wird die Implementierung der besprochenen Erweiterungen und Konzeptintegration in einer prototypischen Entwicklung erklärt, woraufhin eine Evaluation im Bezug auf die Anforderungen und aufgetretene Probleme folgt. Der letzte Abschnitt fasst die Ergebnisse dieser Arbeit zusammen und bietet einen Ausblick auf etwaig folgende Forschung.

2 Grundlagen

Die Verbindung von Konzepten der CPS mit der Rückkopplung in Prozessen erfordert das Einordnen der verschiedenen Domänen in einen größeren Kontext. Zum Verständnis der im späteren Verlauf erörterten Erweiterungen bestehender Technologien, ist die Klärung von Grundlagen essentiell. Dieses Kapitel vermittelt einen Überblick zu den wichtigsten in dieser Arbeit angesprochenen Themen.

2.1 Das Internet der Dinge und cyber-physische Systeme

Die zunehmende Verbreitung und Miniaturisierung, sowie der stetige Anstieg von Komplexität erfordern neue Konzepte zur Steuerung und Kontrolle moderner Computersysteme. Ende des vergangenen Jahrhunderts sprach der Informatikwissenschaftler Mark Weiser von der bevorstehenden Allgegenwart dieser Systeme und deren Verweben in die Umgebung unseres täglichen Lebens [Wei91]. Acht Jahre später prägte Kevin Ashton den Begriff des IoT nach eigener Aussage in einem Vortrag [Ash09]:

> *I could be wrong, but I'm fairly sure the phrase „Internet of Things" started life as the title of a presentation I made at Procter & Gamble (P&G) in 1999.*

Weiser und Ashton trugen früh zur Gestaltung der Idee des IoT bei, welche die Omnipräsenz von Dingen oder Objekten beschreibt. Diese sogenannten *Things* sind durch definierte Adressierungsschemata, z. B. mit Hilfe eines *Uniform Resource Identifier (URI)*[1], in der Lage miteinander zu kommunizieren und zu kooperieren um gemeinsame Ziele zu erreichen [AIM10].

[1] www.w3.org/Addressing/#background

Internet der Dinge. In einem globalen Kontext handelt es sich um ein weltweites Netzwerk von miteinander verbundenen eindeutig adressierbaren Objekten, kommunizierend auf der Basis von standardisierten Protokollen [CAB08].

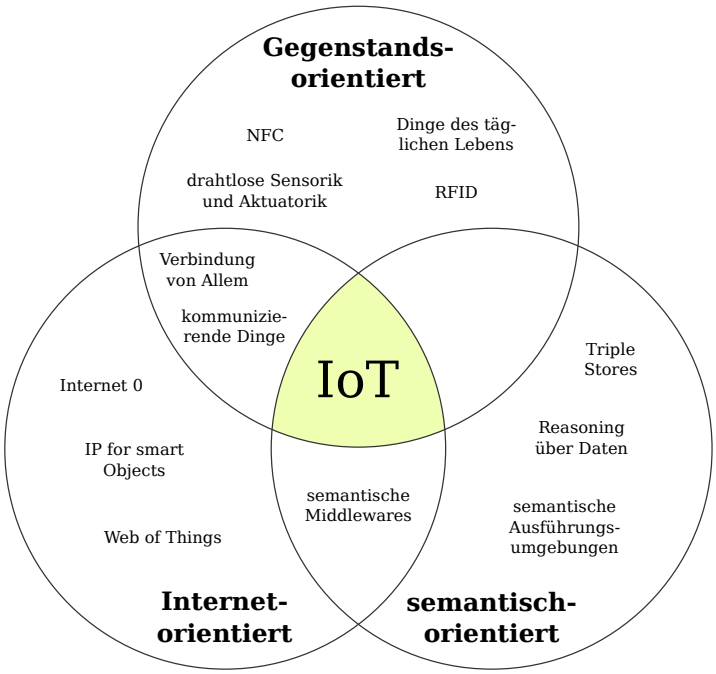

Abbildung 2.1: Das IoT als Schnitt versch. Konzepte und Perspektiven (nach [AIM10])

Weiterhin ist der Begriff durch verschiedene Mengen von Technologien und Protokollen aufteilbar. Die Schnittmenge zwischen Internet-zentrierten, Gegenstandsorientierten und semantischen Paradigmen und Technologien bildet demnach, wie durch Abbildung 2.1 dargestellt, das IoT [AIM10]. Viele dieser Technologien sind heute schon Teil bestehender Infrastrukturen und lassen sich zu einem Netzwerk von Sensoren und Aktuatoren verbinden. Anwendungen solcher umfassender Systeme finden sich unter anderem in der Gebäudeautomation oder Logistik. Sehr spezifische, in der Forschung aktuelle Themen sind *Ambient Assisted Living (AAL)*, *Enhanced Learning* und *E-Health*. Letzteres birgt neben der Identifikation und dem Erfassen des Status eines Patienten, das Messen von Echtzeitdaten, sowie Aspekte der Telemedizin [1] [AIM10].
In der Fertigungstechnik findet sich immer häufiger den Begriff der *Industrie 4.0*. Diese sogenannte vierte industrielle Revolution beschreibt den Paradigmenwechsel hin zu einer stärkeren Vernetzung neuartiger, interaktiver Produktionstechnik, in der eingebettete Steuerungen direkt miteinander kommunizieren und damit zu CPS werden [Rus13].

[1] www.bmg.bund.de/glossarbegriffe/t-u/telemedizin.html

Cyber-physische Systeme binden eingebettete Steuerungen, Sensoren, die Objekte des IoT im Allgemeinen, in Systeme von Systemen ein und ermöglichen ein Orchestrieren von Diensten auf der nächsthöheren Ebene der Abstraktion. Es sind physische und ingenieurwissenschaftlich konstruierte Produkte, deren Operationen durch einen für Berechnungen, drahtlose und -gebundene Kommunikation zuständigen Kern überwacht, koordiniert, kontrolliert und integriert werden [Raj+10]. Während unter physischen Systemen grundsätzlich jene natürlichen Ursprungs verstanden werden, bestimmt der Präfix *cyber* das künstliche, technische Wesen von CPS. Außerdem kombinieren sie Software, Objekte des IoT und Physik, agieren eigenständig und arbeiten kooperativ. Oft handelt es sich um autonome Komponenten, die ursprünglich unabhängig voneinander entwickelt wurden, um spezifische Aufgaben zu lösen [SS12].

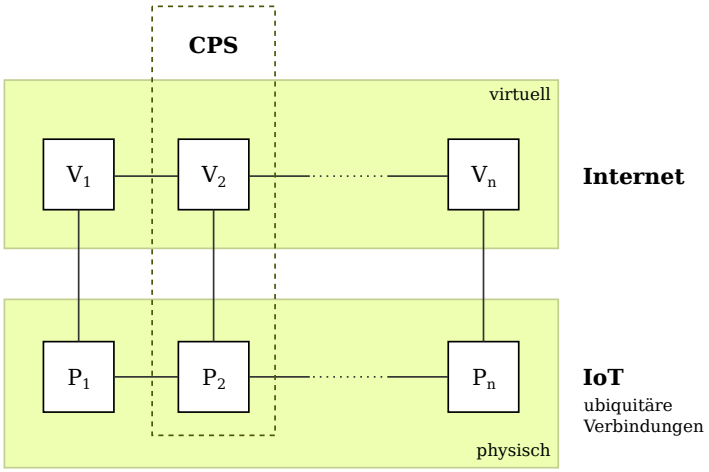

Abbildung 2.2: Relation der Entitäten des IoT bzw. CPS (nach [Che10])

Virtuellen Entitäten, zum Beispiel in Form von Web-Services, und die physischen Dinge des IoT werden zu CPS verbunden, veranschaulicht durch Abbildung 2.2. Die Heterogenität der Schnittstellen dieser Entitäten erschwert zunehmend die Handhabbarkeit der Komposition. Hinzu kommt steigender Kostendruck, der Wunsch nach ressourcenschonendem Verbrauch und die ständig steigenden Anforderungen an Kapazität und Durchsatz. Auch fortschreitende Verkleinerung der Bauelemente erweitert nicht nur den Horizont von Anwendungsfällen, sondern erfordert immer häufiger eine drahtlose Kommunikationsform bei stetig steigender Bandbreite [Raj+10]. Diese durch Luftfahrt, Gesundheitswesen und Industrieautomation gezogene Entwicklung führt unweigerlich zu einer auf Systemebene nicht beherrschbaren Komplexität.

2.2 Prozessmanagement und -modellierung

Die notwendige Abkehr vom reinen Programmieren komplexer Systeme, hin zur komponentenorientierten Komposition von Softwaresystemen, ist nicht der einzige Trend in der Softwaretechnik. Van der Aalst et al. benannten drei weitere, seit den sechziger Jahren anhaltende Entwicklungen. Während die frühe Zeit der Informationsverarbeitung durch Daten getrieben war, spielt Prozessorientierung heute eine stetig größer werdende Rolle. Außerdem verdrängen organisches Wachstum, Agilität und Redesign sukzessive die detaillierter Planung im Vorfeld. Immer häufiger werden bestehende Systeme integriert, wobei die Anzahl der am Reißbrett entworfenen weiter sinkt [AHW03]. Neue, sowie bereits bestehende Komponenten unterliegen damit einer starken Dynamik und müssen für die Integration in bestehende Infrastrukturen vorbereitet sein. Sie sind meist Teil einer umfassenden Architektur, innerhalb derer Abläufe und deren einzelne Schritte durch geeignete Modellierung beschrieben[1], und durch spezifisches Management kontrolliert werden. Ein solcher Ablauf wird im Folgenden Prozess genannt, wobei dessen Prozessschritte allgemein entweder einer Aktivität oder einer Aufgabe entsprechen.

Prozesse repräsentieren eine Menge von Prozessschritten die durch eine unidirektionale Ordnungsrelation miteinander verbunden sind, welche die Reihenfolge der Ausführung beschreibt [Sei+13]. Der Entwurf geschieht meist grafisch; fokussiert sind strukturierte Aktivitäten und Aufgaben, die möglichst viele Fälle abdecken müssen [AHW03]. Auf das operationale Geschäft von Firmen fokussiert sind Geschäftsprozesse Erweiterungen von Prozessen um Akteure wie Lieferanten, Kunden und unterschiedliche Abteilungen. Ein einfaches Beispiel der Modellierung eines solchen Prozesses in der, im späteren Verlauf erläuterten Sprache *Business Process Model and Notation (BPMN)*, mit Start, Ende, den Prozessschritten und Transitionen zeigt Abbildung 2.3.

Abbildung 2.3: Beispielhafte Morgenroutine mit BPMN 2.0

[1] *Programmieren im Großen* [DK75]

Business Process Management (Geschäftsprozessverwaltung) ermöglicht Design, Ausführung, Kontrolle und Analyse operationaler Geschäftsprozesse durch geeignete Techniken, Methoden und Software, wobei Menschen, Organisationen, Anwendungen, Dokumente und andere Quellen von Informationen eingebunden werden [AHW03]. Dieses sich durch alle Schichten ziehende Konzept kann von der fachlichen Perspektive auf das System in einem Prozessmanagementsystem gekapselt werden.

Business Process Management Systems (BPMS) sind durch explizites Design getriebene, generische Softwaresysteme, die operationale Geschäftsprozesse verwalten und ausführen. Generisch ist ein solches System im Sinne der Möglichkeit der Modifikation des unterstützten Prozesses [AHW03]. Das Modellieren eines Prozesses erfordert die Beschreibung dessen in einer für Menschen verständlichen, gleichzeitig durch Maschinen interpretierbaren Sprache. Neben dem Festlegen der verfügbaren Elemente trifft sie Aussagen über die Komposition zu einem Prozess [Sei+13]. Wie die *Unified Modeling Language (UML)*[1], haben diese Formalismen die Aufgabe die Kommunikation zwischen allen in den Prozess involvierten Personen zu vereinfachen. Vom Business Analyst, als Verantwortlichem für das Design des Geschäftsprozesses, über den Entwickler, mit der Aufgabe die technischen Randbedingungen zu schaffen, bis zu dem den Prozess analysierenden Business Manager, existiert ein einheitliches Kommunikationsmittel.

Business Process Model and Notation ist eine seit 2005 in der Hand der *Object Management Group (OMG)*[2] befindliche Spezifikationssprache, die 2010 in der Version 2.0 veröffentlicht wurde [Obj11]. Sie stellt grafische Symbole zur Modellierung von Geschäftsprozessen zur Verfügung und besitzt eine Repräsentation in der *Extensible Markup Language (XML)*. Des Weiteren beinhaltet sie die Möglichkeit der eingeschränkten Spezifikation formaler Ausführungssemantik und integriert die Beschreibung der Beteiligung von Menschen.

Web-Service Business Process Execution Language (WS-BPEL) ist ein Standard der *Advancing open standards for the information society (OASIS)*[3] zur Beschreibung des Verhaltens von Geschäftsprozessen, der Web-Service-Aufrufe als ausschließlichen Informationskanal nutzt und seit 2007 in der Version 2.0 vorliegt [OAS07]. Wie bei der BPMN auch ist das Austauschformat die XML. Die Interpretation einer solchen Beschreibung und damit das tatsächliche Ausführen des Prozesses

[1] www.uml.org
[2] www.omg.org
[3] www.oasis-open.org

erfordert eine *Workflow Engine*, d. h. ein System, welches mögliche Eingaben entgegennimmt, die Aufrufe der Web-Services durchführt, auswertet und den Status des Prozesses aktualisiert. Seiger et al. entwickeln ein *Process Execution Environment for Ubiquitous Systems (PROtEUS)* als eine solche Ausführungsumgebung für CPS [SHS15]. Jedoch fordert die durch Seiger aufgezeigte Inkompatibilität von CPS mit den vorhandenen Spezifikationen auch eine angepasste Modellierungssprache, die durch ein Metamodell definiert wird [Sei+13].

2.3 Metamodell zur Formalisierung von Prozessen

Die Anforderungen an Prozesse und deren Softwaresysteme ändern sich durch verteilte, komplexe Systeme wie CPS stark. Aufgrund der damit einhergehenden Inflexibilität und dem Mangel an Ausdrucksfähigkeit etablierter Modellierungssprachen, entwickelten Seiger et al. ein Metamodell zur Beschreibung von Prozessen in intelligenten cyber-physischen Umgebungen [Sei+13].

Metamodelle definieren eine Menge valider Modelle und ermöglichen deren Transformation, Serialisierung und Austausch [AZ06]. Sie beschreiben Modelle auf einer inhaltlich höher liegenden Ebene, was durch den Präfix *Meta* (griechisch für *über* beziehungsweise *neben*) hervorgehoben wird. Wie Modelle sind sie beschränkte *Abbildungen* der Wirklichkeit und repräsentieren ein natürliches oder künstliches Original [Sta73]. Meta-Metamodelle, wie in Abbildung 2.4, sind dementsprechend solche, die den Sachverhalt eines Metamodells auf einer darüberliegenden Ebene beschreiben. Diese Art der Modell-Hierarchie wurde durch die OMG als *Meta Object Facility (MOF)* formalisiert [Obj15]. Mindestens zwei Meta-Schichten verbinden unterschiedliche MOF-konforme Modelle durch Transformationen und erlauben die Modellierung von Konzepten und deren Instanzen. Vier Meta-Ebenen, die durch *Instanz-von*-Relationen verbunden und in Tabelle 2.1 beschrieben werden, sind Teil der Spezifikation [Obj15].

Beginnend bei M3, definieren Seiger et al. ein Meta-Metamodell für Prozessmodelle, dargestellt durch ein UML-Klassendiagramm in Abbildung 2.4. Das generische `ProcessModelElement` besitzt demnach die Instanzen `ProcessComponent` und `Relation`, die zum Beispiel einen Prozessschritt und dessen Übergang in den nächsten repräsentieren. Repräsentationen dieser Form werden durch die M2-Ebene, das Metamodell für Prozesse als Instanz des Meta-Metamodells, beschrieben. Vorteilhaft ist eine solche Modellierung im Bezug auf Eigenschaften der Objekt-Orientierung

Tabelle 2.1: Hierarchie der Metamodellierung

M0	konkrete Instanzen abstrakter Konzepte und Daten der realen Welt z. B. der Film *Idiocracy*
M1	Domänenmodelle die Daten und Strukturen der M0-Ebene definieren z. B. das Konzept *Film*
M2	Metamodelle, die den Aufbau der Domänenmodelle vorgeben z. B. besitzt das Konzept des Films das *Attribut* Name
M3	Meta-Metamodelle, die Konzepte der M2-Ebene formalisieren z. B. die Definition eines Attributes

(OO). So wird das Konzept von Komponenten und Abgeschlossenheit von Entitäten übernommen (vgl. *Single-Responsibility-Prinzip (SRP)*). Weiterhin existieren in einem Prozess Schnittstellen, welche die Anforderungen beschreiben und Interna verbergen (vgl. Datenkapselung) [Sei+13]. Die in der OO verfügbaren Arten von Relationen, sind Bestandteil des Metamodells und damit auch der Prozessmodellierung. Jedes `ProcessModelElement` kann Teil einer Komposition, Aggregation, oder Vererbungshierarchie sein, die als `Relation` im Meta-Metamodell vertreten sind. Der

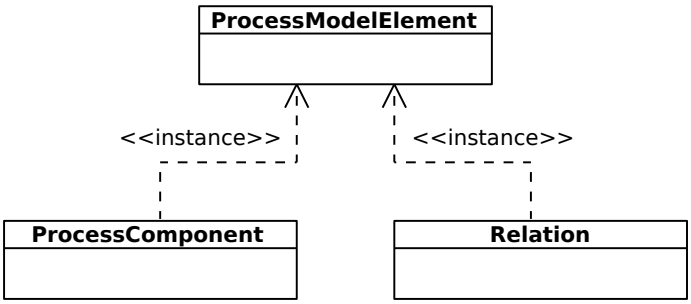

Abbildung 2.4: Meta-Metamodell für Prozesse (nach [Sei+13])

Prozessschritt (`ProcessStep`) ist das zentrale Element des Metamodells von Seiger et al., dessen teilweiser Aufbau durch Abbildung 2.5 verbildlicht wird. Er repräsentiert eine auszuführende Aufgabe oder Aktivität und kann in atomarer Form vorliegen (`AtomicStep`), oder wie im Falle des eigentlichen Prozesses als zusammengesetzter Schritt aus mehreren Teilschritten bestehen (`CompositeStep`). Weiterhin kann der Prozess selbst als Schritt gesehen werden. Atomare Schritte können, wie im Falle der `OR`-Relation, oder als bedingte Anweisungen und Verzweigung (`IF`), logische Operationen abbilden. Auch besteht die Möglichkeit, einen externen Service

aufzurufen (`ServiceInvoke`) oder Daten zu manipulieren (`DataManipulation`). All diese Spezialisierungen werden durch die Vererbungsbeziehung mit dem atomaren Prozessschritt ermöglicht. Um einen Prozess tatsächlich ablaufen lassen zu können, ist im Modell die `Transition` definiert. Sie repräsentiert eine unidirektionale Verbindung zwischen den Schritten und ermöglicht den geordneten Ablauf des Prozesses. Die Verknüpfung zwischen Transitionen und den Prozessschritten geschieht über Ports. Das Modell unterscheidet zwischen Daten- und Kontrollfluss, was durch die Typen `DataPort` und `ControlPort` abgebildet wird. Datenelemente ermöglichen den Daten-Port eines konsumierenden oder produzierenden Prozessschrittes zu typisieren. Sind alle eingehenden Ports eines Schrittes im aktivierten Zustand, so wird dieser ausgeführt. Das Modell definiert Ereignisse in Form von atomaren Schritten, die ihrerseits spezifische Situationen innerhalb des Prozesses induzieren. Sie führen zu weiteren Ereignissen oder stoßen den Beginn neuer Prozesse an. In Kombination mit dem Port-Mechanismus, wird die lose Kopplung komplexer Softwaresysteme unterstützt. Eine weitere spezielle Ausprägung des atomaren Schrittes ist der `ProcessSlot`. Er existiert für den Fall der Nichtverfügbarkeit der Implementation von Prozessschritten in der Phase der Modellierung. Als Platzhalter ermöglicht er den Austausch gegen einen konkreten Schritt zur Laufzeit. Dieser geschieht durch die Suche nach einer Implementation in einem *Repository*, d. h. einer zentralen Datenablage, anhand der typisierten Ports und deren Namen. Eine Ressource (*handling entity*) ist verantwortlich für das Durchführen eines oder mehrerer Prozessschritte. Sie repräsentiert ein Gerät, einen Service, oder einen Menschen als Teil des Systems und kann manuell, auf Instanz- beziehungsweise Typ-Ebene, oder aufgrund von Laufzeitfaktoren festgelegt werden (`ProcessSlot`).

Die Implementierung dieses Metamodells auf Basis des *Eclipse Modeling Framework (EMF)*[1], sowie die integrierte Entwicklungsumgebung für Prozesse in CPS, wird in dieser Arbeit verwendet.

2.4 Die MAPE-K Rückkopplungsschleife

In der Umgebung eines modernen Softwaresystems, steigen Komplexität und Unsicherheit kontinuierlich mit dessen Evolution. Infolgedessen müssen viele der operationalen Handlungsentscheidungen, die vormals statisch, also beim Entwurf des Systems getroffen wurden, nunmehr dynamisch erkannt und festgelegt werden. Dieser Veränderung, unter anderem von Kontextfaktoren, wird vermehrt mit Selbstadaption begegnet [Bru+09].

[1] www.eclipse.org/modeling/emf

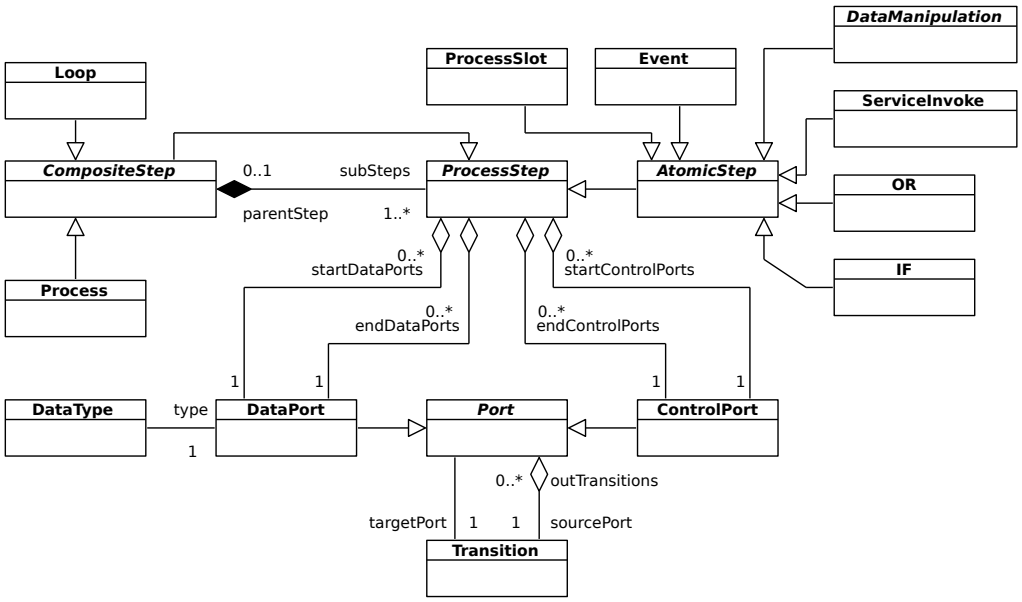

Abbildung 2.5: Ausschnitt des Metamodells für Prozesse (nach [Sei+13])

Autonomes Computing. Zur Kontrolle der eigenständigen Adaption des Managements von Soft- und Hardwaresystemen, wird *autonomes Computing* eingesetzt. Dieses hält verstärkt Einzug in moderne Entwicklungsprozesse. Der Begriff *autonom* kommt ursprünglich aus der Humanbiologie. Er beschreibt dort die Regulation des Körpers durch das zentrale Nervensystem. Es überprüft beispielsweise die Herzfrequenz und den Blutzucker und hält außerdem die Körpertemperatur konstant. Diese Form der Adaption geschieht ohne das Bewusstsein des Menschen. Sie ist vergleichbar mit der automatischen Antizipation von Anforderungen und der Problemlösung mit minimalem, manuellem Zutun. Selbstmanagement, beziehungsweise autonome Regulation, resultiert aus der Fähigkeit Aktionen aufgrund veränderter Situationen selektieren zu können. Angewandt auf das Beispiel der Biologie, hat der Mensch die Möglichkeit Aufgaben höheren Wertes im Bezug auf spezifische Ziele zu fokussieren [IBM06]. Das auch diesem System zugrunde liegende Konzept ist die Rückkopplungsschleife, welche aus vier zentralen Phasen besteht. Das Beobachten und Erfassen von Eigenschaften des autonomen Systems und dessen Kontext und Umgebung bildet den ersten Schritt. In einem zweiten werden sowohl historische als auch Echtzeitdaten analysiert. Neben den gesammelten Werten der ersten Phase, können auch externer Quellen hinzugezogen werden. Nach Abschluss der Analyse aller Informationen wird das weitere Handeln des Systems unter der Berücksichtigung einer Menge von Zielen, Strategien und Regeln geplant. Die vierte Phase ist verantwortlich für die Umsetzung dieser Pläne [Vid+15].

Bei der Erforschung von autonomen Computing postulierte IBM eine Instanz jener phasenbasierten Regelkreise [IBM06]. Das Akronym *MAPE-K* beschreibt neben den Phasen *Monitor*, *Analyze*, *Plan* und *Execute* eine *Knowledge*-Komponente, die als zentrale *Wissensbasis (WB)* fungiert. In Form eines Repository, beinhaltet diese Symptome, Regeln, Änderungsanfragen und -strategien, sowie Logs und Metriken. Im weiteren Verlauf dieser Arbeit wird unter einem *Log* die Quelle protokollarisch abgelegter Informationen zu einem Prozess verstanden. Der im Folgenden beschriebene Aufbau des Gesamtkonzepts ist in Abbildung 2.6 schematisch dargestellt.

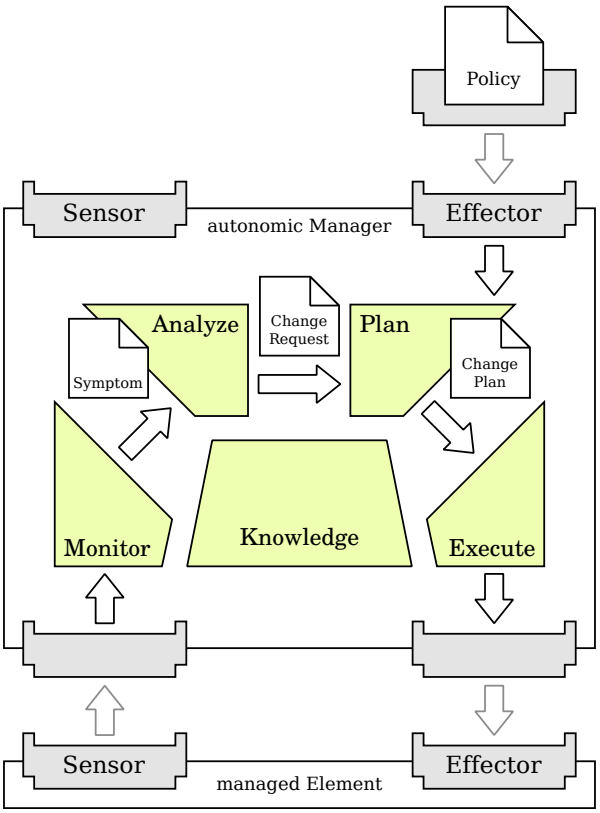

Abbildung 2.6: Schema der MAPE-K Rückkopplungsschleife nach [Bru+09; IBM06]

Autonomes Element. Das durch IBM vorgestellte autonome Element, besteht aus zwei zentralen Bausteinen. Die Implementierung der intelligenten Kontrollschleife ist der *autonomic Manager*. Er sammelt Messwerte des managed Element, beziehungsweise aggregiert Informationen verschiedener Quellen über vergangene und gegenwärtige Zustände. Als *managed Element* wird die zu kontrollierende Kom-

ponente, bestehend aus Sensoren (*Sensors*) und Aktuatoren (*Effectors*), bezeichnet. Diese wird durch eine standardisierte Schnittstelle bezüglich der aktuellen Zielvorgabe vom autonomic Manager angepasst. Kollaboration, sowie Daten- und Kontrollintegration zwischen autonomen Elementen zeichnet diese Schnittstelle aus [Bru+09]. Weiterhin besteht die Möglichkeit der transparenten Komposition von autonomen Elementen zu einer hierarchischen Struktur [IBM06]. In der Arbeit von Dautov et al., repräsentieren managed Elements Soft- und Hardwareressourcen, die durch die Verbindung zum autonomic Manager um autonomes Verhalten erweitert werden [Dau+13]. Cloud-Plattformen, Web-Server, das Betriebssystem oder die ausgeführte Applikation sind Beispiele solcher Ressourcen. Das Ersetzen eines Web-Servers, oder der Neustart eines Betriebssystems werden nach Dautov et al. als *grobkörnige* Veränderungen bezeichnet, wohingegen das Anpassen eines Konfigurationsparameters als *feinkörnig* beschrieben wird. Die von Vidal et al. genannten Phasen einer Kontrollschleife [Vid+15], werden durch MAPE-K wie folgt abgebildet. Aggregation und darauf folgendes Filtern von Details des managed Element und dessen Kontext, erfolgt durch die *monitor*-Funktion und wird nach Dautov et al. in zwei Arten unterschieden [Dau+13]. Wird das managed Element nicht verändert, so wird dies als passives Monitoring bezeichnet. Es ist kontextbezogen und daher *nonintrusive* oder berührungsfrei. Beim aktiven, *intrusive* Monitoring, müssen bei Design und Implementation des Systems, Entry-Points am managed Element berücksichtigt werden. Ein Beispiel für solch eine Schnittstelle ist das *Advanced Programming Interface (API)*. Über dieses erfolgt dann die Aggregation der zu überwachenden Eigenschaften. Relevante Ereignisse (*Events*) in diesem Strom von Daten werden in der WB abgelegt. Unter diesen Metriken und Messwerten verbergen sich *Symptome*, die korrigierende Aktionen nach sich ziehen können [Vid+15]. Die Analyse historischer und Echtzeitdaten erfolgt durch die *analyze*-Funktion. Korrelation von Informationen und Modellierung komplexer Situationen, wie Prognosen anhand von Zeitreihen, gehören zu dieser Phase. In der WB abgelegte Events werden mit den resultierenden verglichen, Symptome diagnostiziert und für die spätere Referenzierung persistiert. Regeln und Event-Muster für Symptome können sowohl *statisch*, beim Entwurf des Systems, als auch *dynamisch* auf Basis neuer Informationen festgelegt werden [Dau+13]. Es besteht die Möglichkeit des Lernens aus vergangenen und die Vorhersage von Zuständen künftiger Situationen. Änderungsanfragen, sogenannte *Change-Requests*, werden hier gebildet und ausgelöst. Unter Berücksichtigung von Zielen (*Goals*), Zielvorgaben (*Objectives*) und Regeln (*Policies*), wird das Planen des weiteren Handelns durch die Interpretation der Symptome in der *plan*-Funktion realisiert. Im Gegensatz zu einem Goal, welches abstrakt, generisch und qualitativ formuliert vorliegt, stellt ein Objective den konkreten, quantitativ zu erreichenden Wert dar. Das Ausführen der erstellten Änderungsstrategie (*Change-Plan*) durch Aktuatoren und externe Services, wird mit der *execute*-Funktion an das managed Element übertragen.

3 Forschungsstand

3.1 Adaption bei Service-Aufrufen

Prozesse mit einer hohen Anzahl von Aktivitäten, akquirieren eine Vielzahl von Ressourcen, müssen hochverfügbar sein und benötigen, aufgrund ihrer unüberschaubaren Komplexität, einen Mechanismus zur automatischen Anpassung und Fehlerkorrektur. Sie besitzen Informationen über ihre Ausführung die automatisch generiert werden. Protokolle, beziehungsweise Logs, beschreiben den Pfad durch das Prozessmodell bezüglich Zeit und Aktivität, sowie zahlreiche Metadaten. Sensordaten und Kontextinformationen reichern den Prozess um den Zustand seiner mittelbaren und unmittelbaren Umgebung an [JMM12]. Metriken, wie CPU-Auslastung, verfügbarer Hauptspeicher und Netzwerkdurchsatz stellen Daten zu einem konkreten System [Sch+13]. Das *Business Activity Monitoring (BAM)*, als ein wichtiger Bestandteil der Prozessanalyse, beschreibt die Diagnose von Prozessen anhand jener Daten [AHW03]. Dessen Ergebnisse verhelfen verteilten Systemen zur Selbstadaptivität, zum Beispiel in der Cloud.

Vienna Platform for Elastic Processes (ViePEP). Schulte et al., entwickelten ViePEP als ein BPMS für elastische Prozesse innerhalb einer Cloud-Umgebung [Sch+13]. Die Elastizität dieser, resultiert aus der Fähigkeit dynamischer Ressourcenallokation zur Anpassung an veränderliche Arbeitslast. Mittels einer Shared-Memory Architektur und gegebenen Attributen der *Quality of Service (QoS)*[1], ist das System in der Lage den künftigen Ressourcenbedarf der Prozesse zu antizipieren [Hoe+13]. Aufgrund des Ergebnisses des *Reasoners*, einem Inferenzmechanismus für logische Konsequenzen, werden Maßnahmen zur Adaption, wie das Starten zusätzlicher Service-Instanzen, ergriffen.

[1] Dienstgüte aus Sicht des Anwenders/Nutzers

Vienna Dynamic Adaption and Monitoring Enviroment for WS-BPEL (VieDAME). Als ähnliches Konzept bezogen auf externe Services, entwickelten Moser et al. mit VieDAME, ein System für die Überwachung und dynamische Adaption von Services in WS-BPEL Prozessen [MRD08]. Das Monitoring dieser geschieht auf Basis von QoS-Attributen wie Verfügbarkeit und Antwortzeit. VieDAME besteht im Kern aus dem auszuführenden Prozess mit seinem bestimmten Kontrollfluss und den über das Internet verteilten Service-Aufrufen. Ein Repository hält Referenzen und Metainformationen zu allen verfügbaren Diensten. Diese können als ersetzbar markiert und bei sinkender QoS durch eine Alternative ausgetauscht werden. Unterschiedliche Strategien befähigen das System zum Austausch, auch ohne den laufenden Prozess zu verändern. Die aspektorientierte Programmierung (AOP) ermöglicht die Transformation von Nachrichten, um syntaktische und semantische Abweichungen der Schnittstellen auszugleichen. AOP ist ein Paradigma, welches explizite Mechanismen zum Erfassen der Struktur von Querschnittproblemen (*Crosscutting Concerns (CCC)*) bereitstellt und dadurch die Modularität und Trennung von Zuständigkeiten (*Separation of Concerns (SoC)*) bei Software-Design und -Entwicklung verbessert [KH01]. Das Management der verfügbaren Services, die Parameter für QoS-Attribute und deren Überwachung, sowie die Transformationskriterien sind über ein *Graphical User Interface (GUI)* bereitgestellt. Durch die Implementation eines Adapters, ist es möglich die im Prototyp des Projekts verwendete WS-BPEL-Engine auszutauschen.

3.2 Überwachung durch Continuous Monitoring

Software wird vor dem Produktiveinsatz auf Korrektheit überprüft. Diese Validierung findet statisch und beschränkt auf ein Teilsystem statt, wodurch die Integration mit dem Gesamtsystem gesondert überprüft werden muss. Im Umfeld von Service-orientierten, verteilten Systemen ergibt sich weiterhin das Problem der dynamischen Veränderung von Topologie, wodurch die Anzahl der Testfälle jeder Einschränkung entbehrt. Baresi und Guinea entwickelten ein Konzept zur Laufzeit-Validierung mittels *continuous Monitoring*, welches die WS-BPEL um den Aspekt der passiven Überwachung ergänzt [BG05]. Die externen Regeln zum Monitoring werden beim Bereitstellen an den WS-BPEL-Prozess gebunden. Dadurch ergibt sich eine strikte Trennung von Geschäfts- und Kontroll-Logik, wodurch SoC sichergestellt wird. Regeln werden außerdem mit konkreten Elementen, beziehungsweise Aktivitäten des Prozesses assoziiert. Die Spezifikation der Zusicherungen (Assertions) im Bezug auf die Ausführung von Prozessschritten geschieht mittels der *Web-Service Constraint Language (WS-CoL)*. Diese Sprache zur formalen Definition von

Beschränkungen (Constraints) erweitert die *Java Modeling Language (JML)*[1] um Konstrukte zur Aggregation von Daten externer Quellen. Verschiedene Monitoring-Grade können über ein GUI eingestellt werden und erlauben die damit einhergehende Priorisierung, die wiederum die tatsächliche Analyse der Ausführungs- und QoS-Daten beeinflusst. Der dem Proxy-Entwurfsmuster entsprechende *Monitoring Manager* ist verantwortlich für die Evaluation der Regeln, die Interaktion mit den konkreten externen Services und den Aufruf der analysierenden Monitor-Komponente für das Auswerten der Beschränkungen. Wenn Regeln, Beschränkungen oder Zusicherungen verletzt werden, leitet er diese Information an den Prozess zurück. Außerdem besteht die Möglichkeit der Ausführung wiederherstellender Aktionen (*recovery actions*), welche in den Regeln definiert werden. In beiden Fällen entsteht eine auf dem Prozess basierende Rückkopplung.

3.3 Verifikation durch Conformance Checking

Viele Konzepte bauen auf den idealen Prozess, welcher erfahrungsgemäß selten der Realität entspricht. Die Verifikation der Überdeckung des ausgeführten Prozesses mit dem Modell wird *conformance Checking* genannt. In diesem Teilgebiet des *Process Mining* wird der in den Logs abgelegte Kontrollfluss mit den im Modell möglichen verglichen und Inkonsistenzen aufgedeckt [DAV12]. Process Mining Techniken dienen der automatischen Extraktion menschenlesbarer Prozess-Modelle, u. a. aus in Logs enthaltenen Informationen [LMM15]. Trotz guter Verfügbarkeit von Logs in modernen Unternehmen, sind protokollarische Aufzeichnungen meist auch ohne jedes *Process-Aware Information System (PAIS)* vorhanden und können derlei Techniken speisen [DAV12].

Überwachen des realen Verhaltens. Rozinat et al. erarbeiteten die Verifikation der Konformität des Prozessmodells, repräsentiert durch ein Petri-Netz, mit der in den Logs abgelegten, geordneten Abfolge der tatsächlich ausgeführten Aktivitäten [RA08]. Petri-Netze sind gerichtete Graphen, die hauptsächlich aus zwei verschiedenen Knotentypen bestehen. *Plätze* repräsentieren des Systems mögliche Zustände und *Transitionen* sind Events oder Aktionen die eine Zustandsveränderung auslösen [LK06]. Rozinat et al. gliederten Konformität in zwei Dimensionen und entwickelten zu diesen quantifizierende Metriken. Weiterhin ist die Lokalisation eines Problems in Modell und Log durch ihre Arbeit möglich.

[1] www.jmlspecs.org

(1) **Fitness**: *Entspricht der beobachtete Prozess dem durch das Modell spezifizierten Kontrollfluss?*

Ein Log und ein Petri-Netz passen zueinander (*fit*), wenn das Petri-Netz jede Spur (*trace*), d. h. Sequenz von Events, generieren, beziehungsweise *parsen*, kann. Da es möglich ist ein Petri-Netz zu konstruieren, welches jedes Event-Log parsen kann und damit eine quantitative Fitness von 1 zustande käme, wird Konformität durch diese Dimension allein nicht impliziert.

(2) **Appropriateness**: *Ist die Beschreibung des Prozesses durch das Modell geeignet?*

Ockhams Rasiermesser[1] dient dieser Eigenschaft als Vorlage und ist demnach nicht ohne die folgende Unterscheidung Quantifizierbar.

a) **structural**: Das einfache Modell ist dem komplizierten vorzuziehen.
b) **behavioral**: Das Modell sollte nicht zu generisch und zu wenig restriktiv im Bezug auf das Verhalten sein.

Log-beschränkte Modellexploration. Das ausschließlich auf ein Log bezogene Messen der Präzision eines Petri-Netz-basierten Prozessmodells, untersuchten Muñoz-Gama et al. [MC10]. Präzision bezieht sich auf die Generizität eines Modells, wobei sie mit der Minimalität des Verhaltens steigt. Mit den Ergebnissen ihrer Arbeit ist es möglich Inkonsistenzen in Log und Modell zu lokalisieren. Durch die Beschränkung der Exploration auf das Log, wird eine Explosion der Menge möglicher Prozesszustände vermieden. Das Konzept der *Escaping Edges* beschreibt Situationen in denen das Modell mehr Verhalten erlaubt als die Event-Sequenz impliziert, wodurch die Präzision vermindert wird. Es bezieht sich auf die Knoten des vereinigten Zustandsraums von Modell und Log.

Die Methode beginnt mit der auf das Log beschränkten Berechnung des Modellverhaltens, in Abbildung 3.1 dargestellt durch den grau hinterlegten Bereich. Die Grenze zum Verhalten des Logs beinhaltet jene Punkte, an denen das Modell abweicht und somit die der Escaping Edges. Durch die Quantifizierung dieser Knoten und deren Frequenz, wird das Maß der Präzision beschrieben. Außerdem können sie auf Inkonsistenzen hinweisen, die in der weiteren Entwicklung des Prozesses berücksichtigt werden müssen. Neben der Lokalisation der Diskrepanzen, ist die Berechnung des kritischen Pfades im Modell ein Nebenprodukt dieser Methode.

[1] Um einen Sachverhalt zu erklären, sollte die Zahl der Entitäten nicht über das notwendige Maß hinaus vergrößert werden [RA08].

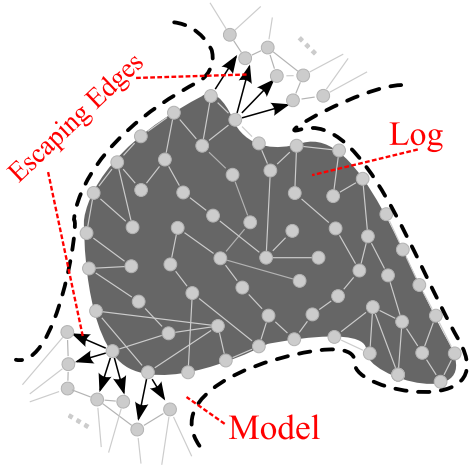

Abbildung 3.1: Prinzip der Escaping Edges (aus [MC10])

Einbeziehen von Daten und Ressourcen. Conformance Checking kann nach De Leoni et al. nicht nur auf dem Kontrollfluss eines Prozesses betrieben werden [DAV12]. In ihrer Arbeit zeigten sie die Möglichkeit auch Datenfluss und Ressourcen einzubeziehen. Beispielsweise ist es bei der Diagnose des Datenfluss möglich, den genauen Wert der Abweichung eines Attributes nachzuvollziehen. Auch kann festgestellt werden, welche Ressourcen oder Aktivitäten dauerhaft die ihnen zugeteilten Rechte verletzen. Die Suche nach einem *Alignment*, dem Abgleich von Prozessmodell und Event-Log, wird durch den A^*-Algorithmus realisiert und identifiziert die Sequenz von Events mit den geringsten Kosten, interpretiert als minimale Abweichung vom Modell und damit größtmöglicher Fitness. A^* ist ein Graph-basierter iterativ vertiefender Suchalgorithmus, der die Kosten für einen Pfad aus der Summe des zurückgelegten Weges und einer Heuristik, verbleibenden Kosten bis zum Zielknoten, zusammensetzt. De Leoni et al. entwarfen ein heuristisches Modell mit sublinearer Berechnungskomplexität und starker Eingrenzung des Problemraums. Sowohl im Bezug auf die Prozessbeschreibungssprache, als auch auf das Format des Logs, ist die Technik generisch.

3.4 Event-getriebene Rückkopplung

Eine der umfangreichsten Herausforderungen in verteilten Systemen ist das Einhalten einer hohen QoS, wofür die kontinuierliche Überwachung der Prozesse notwendig ist. Durch *Complex Event Processing (CEP)* wird diesem Problem mit einer Sammlung von Informationen über die verschiedenen Schritte begegnet. Auf diese

Weise können aufkommende Situationen mit niedriger QoS identifiziert und behandelt werden. CEP erlaubt die Extraktion relevanter Informationen durch das Erkennen von Korrelation unterschiedlicher Events mittels der Analyse des zeitlichen Ablaufs, der Kausalität, sowie der Zugehörigkeit zu einem Strom von Daten [HSD10]. Die Handhabbarkeit der Menge und Heterogenität von Informationen in einem Prozess wird durch CEP stark verbessert und mündet im Konzept des *Event-Driven Business Process Management (EDBPM)*.

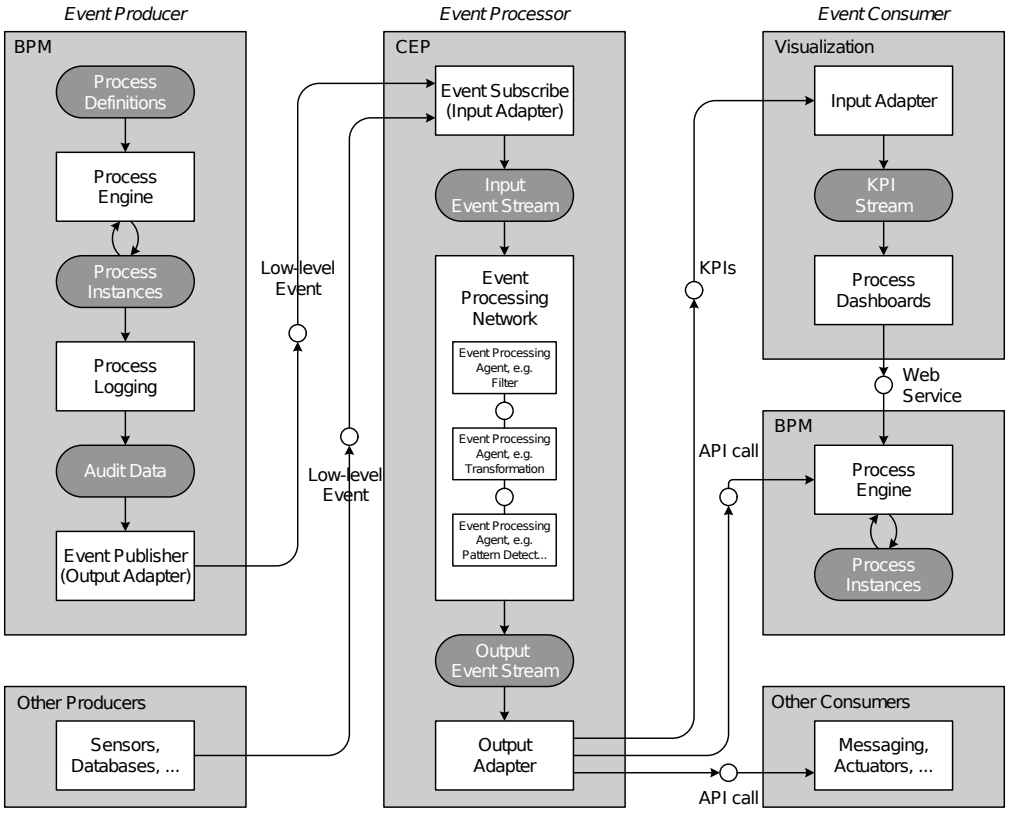

Abbildung 3.2: Architektur eines Systems für EDBPM (aus [JMM12])

Janiesch et al. beschrieben eine Architektur zur Überwachung und Kontrolle von Prozessen durch EDBPM [JMM12], wobei eine CEP-Engine die teilweise Verwaltung des Prozesses aktiv übernimmt. Eine Einschränkung bestehender BPMS besteht ihrer Analyse nach in der geringen Ausprägung von Monitoring und Reporting. So müssen Maßnahmen über eine passives Überwachung hinaus durch externe Erweiterungen integriert werden, da kein explizites Systemkonzept vorhanden ist. Weiterhin besteht nach Janiesch et al. keine Möglichkeit auf Ereignisse laufender Prozesse zu reagieren, wodurch die Reaktion auf Events nur statisch implementiert werden kann. Neben der konzeptuellen Vorbereitung für Kontrollschleifen, sind

Events zentraler Bestandteil und die Verarbeitung dieser geschieht in Echtzeit. Die Architektur, schematisch in Abbildung 3.2 dargestellt, ist nach dem Prinzip der losen Kopplung und SoC in die Komponenten Event-Erzeuger, -Prozessor und -Konsument eingeteilt und erweitert das BAM durch autonome Entscheidungsfähigkeit und Rückkopplungsmechanismen.

Event-Producers sind neben den Logs eines BPMS auch Sensoren und Datenbanken. RFID-Tags können mit der Prozessinstanz verknüpft werden und beispielsweise die Ankunft eines Lieferwagens als Event in den Prozess einfließen lassen. Kontextinformationen wie das Wetter, die Verkehrssituation oder die Bewegung und Position realer Objekte werden Teil des Systems. Auch indirekt zugängliche Informationen über einen Aktienkurs, oder das Anlegen eines neuen Datensatzes können so den weiteren Kontrollfluss direkt beeinflussen. *Event-Processors* beinhalten die eigentliche CEP-Engine und transformieren ein- und ausgehende Daten der Produzenten, beziehungsweise Konsumenten über spezialisierte Adapter. Sie bestehen aus einem Event-Processing Network (EPN), welches Event-Processing Agents (EPAs) untereinander verbindet. EPAs sind reaktive Komponenten, die ähnlich dem *Pipes & Filters* Architekturmuster hintereinander geschaltet und in *filternd*, *transformierend* und *generierend* bezüglich eines Events unterteilt werden. *Event-Consumers* empfangen, visualisieren oder reagieren auf Events des Event-Processors. Auf diese Weise können nicht nur Dashboards, d. h. grafische Visualisierung hoher Informationsdichte, sondern auch Messaging-Services, Aktuatoren und das BPMS selbst die Daten im jeweiligen Sinne nutzen.

3.5 Ziel-basierte Prozesse

Beim BPM liegt der Fokus häufig auf der Prozessausführung allein. Nicht zuletzt wegen der Separation der Modellierung und unzureichenden Möglichkeiten der automatischen Ausführung in realen Umgebungen, werden Modelle oft nur zur Dokumentationszwecken eingesetzt [KK12]. Mit der *Architecture for Business Process optimization (aPro)* wirken Kötter et al. dieser Problematik entgegen und begannen mit der Entwicklung von Prozess-Komponenten für die Modellierung, Ausführung und Überwachung. Auch die Analyse der Monitoring-Daten mittels *Data Mining* und die Adaption des Prozesses durch etwaige Parameteranpassungen ist Teil ihrer Arbeit. Sie erweiterten die BPMN um Elemente für Metriken, *Key Performance Indicators (KPIs)* und Ziele, die in Abbildung 3.3 auch in ihrer Kardinalität bezüglich des Kontrollflusses dargestellt werden. Weiterhin wird das auf XML aufbauende Austauschformat *Process Goal Markup Language (ProGoalML)* zur Automatisierung des Erzeugens und Einrichtens einer Überwachungsinfrastruktur genutzt.

Dieses vermittelt die modellierten Vorgaben zwischen dem Prozessmodell, den Metriken, KPIs und Zielen, den Schemata für Messungen und Ergebnisse, sowie den CEP-Regeln. Die unterschiedlichen Schemata werden als *XML Schema Definition (XSD)* formuliert.

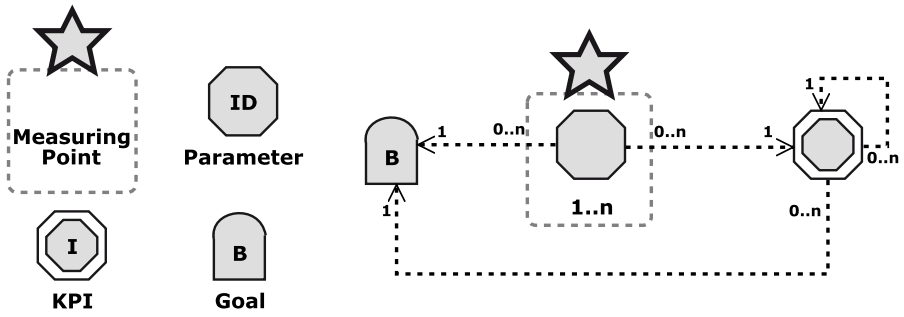

Abbildung 3.3: Modellierungselemente der ProGoalML (aus [KK12])

Aus der Zielvereinbarungen enthaltenden Modellierung des Prozesses, ergeben sich ein ProGoalML- und ein reines BPMN-Dokument, welches in einem Repository für Prozesse abgelegt wird. Die Messpunkte für das Monitoring der Ausführung einer Prozessinstanz, werden aus dem Modell generiert und besitzen je ein *Monitoring-Stub*. Jedes dieser Proxy-Objekte ist für eine spezifische Aktivität zuständig, ob es sich um einen Web-Service Aufruf, oder das periodische Abfragen der Logs handelt. Die Messungen der Stubs werden an einen *Monitoring Web-Service* übergeben, wodurch Unabhängigkeit von der verwendeten CEP-Engine entsteht. Der Monitoring Web-Service wandelt weiterhin die Messung in ein Event und überträgt dieses an die Engine. Um die Ergebnisse der Überwachung zu erhalten, sammelt die CEP-Engine diese Events und ermittelt KPIs und Ziele. Da diese aus Parametern berechnet werden, die potentiell zu mehreren Messpunkten gehörten, ist eine Zusammenfassung der Werte, ähnlich dem *Join* relationaler Datenbanken notwendig. Distinkte *Coverage*-Klassen, beziehungsweise Zusammenfassungen der Messwerte, werden durch einen von Kötter et al. entwickelten Algorithmus ermittelt. Aus den Klassen werden nun Ergebnisschemata und CEP-Regeln generiert. Diese werden durch Abfragen in einer *Event-Processing Language (EPL)* formuliert und liefern ein Ergebnis in dem vorgegebenen Schema. Die EPL, ähnlich der *Structured Query Language (SQL)* für relationale Datenbanken, ermöglicht Abfragen über Strömen von Events. Ein einem Ergebnis enthalten sind die konkreten Werte der Ziele, KPIs und Parameter einer Coverage-Klasse, welche in einem Dashboard visualisiert werden, oder in eine Rückkopplung einfließen können. In der Arbeit von Kötter et al. liegt der Fokus auf der Modellierung und dem Monitoring von Prozessen. Ausführung, Analyse und die Adaption mit *Adaptive Business Process Modeling in the Internet of Services (ABIS)* sind Teil laufender Untersuchungen. Im Kern entwar-

fen sie eine Notation für die Modellierung von Zielen. Außerdem resultierte eine konkrete Transformation der ProGoalML-Dokumente zur Weiterverarbeitung in Web-Services und der CEP-Engine. XML-Schemata für Messungen und Ergebnisse, sowie die CEP-Regeln werden automatisch erzeugt und ermöglichen schlussendlich das Auslösen eines neuen Events.

3.6 Transaktionale Prozesse

Bei verteilten Systemen im allgemeinen, und komplexen Prozessen im Speziellen, erfolgt der parallele Zugriff auf gemeinschaftlich genutzte Ressourcen. Um die verschiedenen Prozessschritte konsistent durchführen zu können, sind Transaktionen notwendig. Diese sind ein Spezialfall des Objekt-relationalen Verhaltensmusters *Unit of Work*, welches Martin Fowler in seinen Enterprise Pattern vorschlug [Fow03]. So werden die Operationen eines Arbeitspakets entweder vollständig oder gar nicht ausgeführt. Tritt innerhalb der Abfolge ein Fehler auf, wird die Transaktion zurückgerollt (*Rollback*), andernfalls wird sie finalisiert (*Commit*). Transaktionen werden meist in relationalen Datenbanken eingesetzt und folgen den Prinzipien *Atomicity, Consistency, Isolation, Durability (ACID)*. Alonso et al. fassten diese wie folgt zusammen [Alo05].

- **Consistency** (Konsistenz) garantiert, dass jede Transaktion die von einem konsistenten Zustand startet, nicht durch eine andere Transaktion im System gestartet, vollständig ausgeführt und ohne Fehler beendet wurde, einen konsistenten Zustand hinterlässt.

- **Isolation** garantiert, dass auch bei konkurrierenden Transaktionen, sich die einzelne verhält, als wäre sie die einzige im System. Isolation erlaubt das parallele Ausführen von Transaktionen, bei gleichzeitigem Wahren von Konsistenz.

- **Atomicity** (Atomizität) stellt sicher, dass eine Transaktion die Datenbank immer in einem konsistenten Zustand hinterlässt. Durch das Konsistenzkriterium ist dies nur erfüllt, wenn die Transaktion vollständig abgeschlossen oder gar nicht durchgeführt wird.

- **Durability** (Dauerhaftigkeit) als die am wenigsten eindeutige Eigenschaft, garantiert, dass nach der erfolgreichen Ausführung einer Transaktion die durchgeführten Veränderungen nicht verloren gehen.

Der Versuch des Übertragens dieser Prinzipien auf die Prozessausführung, mündet in der Idee eines transaktionalen Prozesses [Sch09]. Dieser besteht aus einer partiell geordneten Sequenz von Aktivitäten. Die Art seiner Ausführung garantiert transaktionale Konsistenz für alle Aktivitäten oder eine Teilmenge derer. Sie sind transaktional, wenn sie selbst transaktionale Prozesse oder Transaktionen innerhalb einer Datenbank sind. Bestehen sie aus dem Aufruf eines Services, sind sie nicht-transaktional. Die Frage nach der Umsetzung von Prozessen mit Transaktionen, ist nicht ohne weiteres zu beantworten. Hamadi et al. erklärten die Unangemessenheit der Implementierung des Prozesses in einer einzigen Transaktion folgendermaßen [HBM08]. Wenn gemeinsam genutzte Ressourcen schreibend geändert werden sollen, so müssen diese für andere Schreibzugriffe gesperrt werden. Jedoch sind Prozesse häufig sehr zeitintensiv, wodurch die Sperrung sehr langfristig und die Wartezeit anderer Konsumenten der Ressource untragbar würde. Während der Ausführung einer Prozessinstanz, werden viele unterschiedliche Datenbanken und Services angesprochen, wodurch das Durchsetzen der ACID-Prinzipien einer aufwändigen Koordination bedarf. Aufgrund externer Effekte während der Ausführung von Prozessen, ist das Wahren der Atomizität mit konventionellen Mechanismen zum Zurücksetzen von Transaktionen impraktikabel. Zusammenfassend sind die bei Datenbanken gefestigten Kriterien für transaktionale Prozesse zu restriktiv [Gar+12].

Sagas. Atomizität durch Kompensation, wurde bereits 1987 mit dem Sagas-Modell von Garcia-Molina und Salem beschrieben [GS87]. Wird beispielsweise die fehlerhafte Bestellung eines Artikels zu früh bestätigt, so muss beim Rollback der Transaktion, dieser Schritt umgekehrt werden. Die Bestätigung für jene Bestellung wurde jedoch versandt und ist damit nicht mehr zu widerrufen. Eine Nachricht über die Stornierung muss diesen Schritt kompensieren. Kompensierende Transaktionen bieten demnach ein semantisches *undo* [Por+06]. In Sagas entspricht der Prozessschritt T_i einer eigenständigen Transaktion konventioneller Form und besitzt die kompensatorische Aktivität T_i^{-1}. Ein Prozess mit n Schritten kann nun zwei Wege einschlagen [Por+06].

- $T_0, T_1, \ldots, T_n$ wenn jede Transaktion T_i zum Commit führt
- $T_0, T_1, \ldots, T_{i-1}, T_i^{-1}, \ldots, T_0^{-1}$ wenn die Transaktion $T_i (0 \leq i \leq n)$ einen Rollback auslöst

Dementsprechend ist ein Prozess entweder erfolgreich oder er wurde kompensiert. Bei der Modellierung solcher kompensierender Prozesse, muss jedem relevanten Schritt eine kompensatorische Aktivität zugewiesen werden. Außerdem ist die Trennung von Geschäftslogik und deren Fehlerbehandlung kaum gegeben. Die Prozessausführungssprache WS-BPEL ist mit Elementen zu Kompensation und Fehlerbehandlung ausgestattet.

Semi-Atomizität. Die Möglichkeiten des Sagas-Modells sind begrenzt auf Abschluss oder Kompensation des Prozesses. Es ist häufig sinnvoll, einen teilweisen Rollback in langlaufenden Prozessen anzustoßen. Diese Semi-Atomizität für flexible Transaktionen, wird durch alternative Pfade im Prozessmodell ausgedrückt. Nach Zhang et al. ergeben sich, bezogen auf deren Transaktionalität, drei Typen von Prozessschritten [Zha+94].

- **compensatable** (kompensierbar): Schritt besitzt eine kompensatorische Aktivität
- **retriable** (wiederholbar): der Erfolg ist durch endliches Wiederholen des Schrittes garantiert
- **pivotal** (zentral): Schritt ist weder compensatable noch retriable

Es ergeben sich weitreichende Implikationen, die auch das Modellieren von Prozessen beeinflussen [Alo05]. Wurde ein zentraler Schritt ausgeführt, muss ein terminierender Pfad existieren. Die Schritte vor einem solchen müssen alle kompensierbar sein. In der Konsequenz steigt die Komplexität, der Typ eines jeden Schrittes muss identifiziert werden und nur die korrekte Kombination erzeugt ein valides Modell. Vorteilhaft an Semiatomizität ist die Flexibilität im Bezug auf Fehlerbehandlung. Außerdem ist es möglich Prozesse automatisch zu Validieren.

X/Open XA. Das Involvieren eines Prozessdesigners in die Modellierung von Transaktionsattributen verursacht Komplexität über das zu lösende Domänenproblem hinaus. Notwendige Transparenz, sowie Verteilung von Transaktionen, ist Teil des Open Group[1] Standards X/Open *eXtended Architecture (XA)* [Ope92]. Er bietet ein grundlegendes Konzept für verteilte Transaktionen in Geschäftsprozessen, welches unter anderem durch den *Object Transaction Service (OTS)* in der *Common Object Request Broker Architecture (CORBA)* implementiert wird (vlg. [Alo05]). Die Spezifikation basiert auf einem *Two-Phase-Commit (2PC)*, d. h. dem Finalisieren einer Transaktion in zwei Phasen [Alo05]. Neben den Teilnehmenden Datenbanken (*Participants*), ist ein Koordinator für das Sammeln der Stimmen (*Votes*), in

[1] www.opengroup.org

der ersten Phase, und das Entscheiden über den Commit der Transaktion zuständig. Sind alle Teilnehmer mit der Finalisierung einverstanden, sendet der Koordinator in der zweiten Phase die Nachricht zum jeweilig lokal Commit. Der XA-Standard stellt einige Anforderungen. So muss ein Transaktionskontext vorhanden sein, damit alle Teilnehmer den Urheber einer Operation kennen. Weiterhin benötigt die XA eine Registrierung, sodass der Koordinator den 2PC-Ausführenden ermitteln kann. Grundsätzlich müssen alle Beteiligten das 2PC Protokoll unter der Obhut des Koordinators unterstützen. Wie in Abbildung 3.4 dargestellt, beinhaltet die XA ein

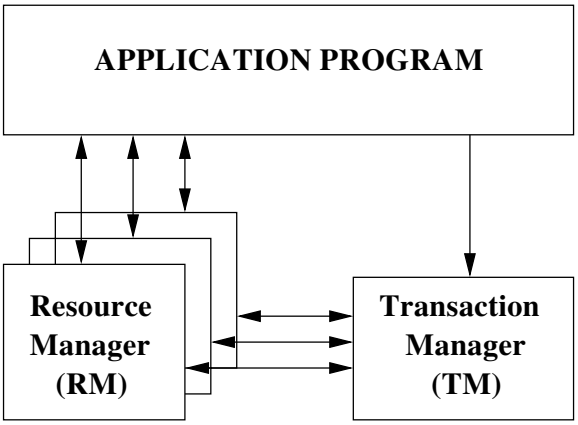

Abbildung 3.4: Beteiligte aus X/Open XA (aus [Alo05])

Application Program (AP), welches mittels des *Transaction Manager (TM)* über dem Start und das Ende einer Transaktion entscheidet. Dieser TM ist Koordinator der verteilten Transaktion. Die *Resource Managers (RMs)* kapseln verwendete Ressourcen und führen die durch das AP angestoßenen Operationen unter der Kontrolle des TM aus. Der XA-Standard definiert, wie der TM und die RMs kommunizieren um transaktionale Konsistenz mittels 2PC zu gewährleisten.

Web Services Transaction. Das Ausnutzen der Transaktionalität von Datenbanken im XA-Standard, lässt sich nicht ohne weiteres auf Web-Services übertragen, wodurch diese als Teilnehmer im 2PC ausscheiden. Um die Auswirkungen einer Aktion in einem verteilten System koordinieren zu können, definierte OASIS einige Protokolle, welche über Datenbanktransaktionalität hinaus gehen [OAS09]. *Web Services Transaction (WS-TX)* überträgt die Prinzipien der XA in den Kontext von Web-Services durch die Generalisierung der Spezifikation [Alo05]. In einem verteilten System gilt es, einen *Single Point of Failure (SPOF)* zu vermeiden. So ist ein einzelner TM ungeeignet für die verlässliche Koordination von Transaktionen, was durch die Unterspezifikation *WS-Coordination* einbezogen wird. Sie stellt ein

generisches Framework mit erweiterbaren Commit-Protokollen für die Koordination verteilter Aktivitäten. Der vom Koordinator erzeuge Transaktionskontext beinhaltet neben der Identifikation, übernommen aus der XA, den *Uniform Resource Locator (URL)* des Web-Services, sowie das verwendete Protokoll. Dieser ist bei einer Nachricht zwischen den Services Teil der Metadaten des *Simple Object Access Protocoll (SOAP)*. Um das Protokoll ohne infrastrukturelle Änderungen austauschen zu können, modularisiert der Standard die XA durch *WS-Transaction*, bestehend aus den Definitionen *WS-AtomicTransaction* und *WS-BusinessActivity*, die das Festlegen der konkrete Protokolle bewerkstelligen. Erstere deckt, ähnlich denen des 2PC, die Interaktionen zwischen den Web-Services ab. WS-BusinessActivity richtet sich vornehmlich an langlaufende Prozesstransaktionen und enthält zwei konkrete Protokolle.

- **Business Agreement with Participation Completion Protocol (BAP)** Stellt eine vereinfachte Commit-Form dar, die einen Abbruch und kompensatorische Aktivitäten unterstützt.

- **Business Agreement with Coordinator Completion Protocol (BAC)** Ähnlich dem BAP mit einem zusätzlichem Schritt für die Vervollständigung der Arbeit mit einer Variante des *Tentative Hold Protocol* (semantischer vom Prozess abhängiger Mechanismus)

3.7 Business Process Simulation

Die inkrementelle Optimierung von Produktionsschritten und Geschäftsprozessen, führt zu einer Form der Evaluation mittels Varianten und alternativer Konfigurationen. Jedoch sind reale Experimente auf Basis dieser Veränderungen meist nicht reproduzierbar, häufig zu teuer oder gar gefährlich [Aal15]. Simulation außerhalb des produktiven Systems bietet einen Einblick in existierende oder künftige Situationen und ist laut Shannon der Prozess der Modellierung eines realen Systems. Mit diesem Modell können Experimente zum Verständnis seines Verhaltens durchgeführt werden. Außerdem bieten diese die Möglichkeit der Evaluation unterschiedlicher Strategien für die Operation des Systems [Sha98]. So ist es mit *Business Process Simulation (BPS)* möglich, Durchlaufzeiten, Dauer oder Kosten eines neuen oder veränderten Prozesses zu identifizieren. Unterstützung von Kapazitäts- oder Einsatzplanung, sowie die Vorhersage von Leistungsdaten bei veränderten Umgebungssituationen wird durch Simulation ermöglicht. Verschiedene Prozessdesigns im Bezug auf bestimmte Kennwerte können miteinander verglichen werden. Selbiges gilt für unterschiedliche Eingangsparameter [Rüc08]. Dennoch können Simulationsstu-

dien sehr zeitaufwendig sein, die Interpretation von Ergebnissen ist häufig schwierig und Beweise gehen aus diesen nicht hervor [Aal15]. Simulationen werden anhand der Veränderung des Systems über die Zeit, sowie der Zahl möglicher Systemzustände klassifiziert [Mat08]. Verändert sich das simulierte System über die Zeit nicht, so ist es *statisch*, andernfalls *dynamisch*. Besitzt das System eine endliche Anzahl von Zuständen, ist es *diskret*, *kontinuierlich* im entgegengesetzten Fall. Es werden drei Paradigmen im Bezug auf den Ablauf der Simulation unterschieden. *Prozessorientiert* sind Simulationen, die ähnlich den Unix-Prozessen parallel operieren. Durch das Warten auf Ergebnisse eines anderen Prozesses und die Synchronisation zwischen diesen, ist dieses Konzept langsam und schwierig zu implementieren. *Aktivitätsorientierte* Simulationen bewegen sich auf der Ebene von Prozessschritten. Die Realzeit wird sukzessive inkrementiert, wobei nicht jeder Zeitpunkt eine Zustandsänderung im System bewirkt. Somit ist dieses Paradigma Ressourcen-intensiv und langsam. *Ereignisbasierte* Simulationen, sind unabhängig von einer Realzeit. Die Simulationszeit wird mit dem Eintreffen eines Events aktualisiert. Diese Arbeit beschränkt sich auf dynamische, Zustands-diskrete, Ereignis-getriebene Simulationsmethoden, im weiteren Verlauf *Discrete-Event Simulation (DES)* genannt.

Business Process Management und DES führen in Kombination zu BPS und erlauben die Bewertung neuer, sowie die Verbesserung bestehender Prozesse durch Simulation. In der Abschlussarbeit *Building an open source Business Process Simulation tool with JBoss jBPM*, beschreibt Bernd Rücker die Kombination von jBPM[1] und DESMO-J[2] zu einem Werkzeug für die Simulation von Geschäftsprozessen [Rüc08]. jBPM ist ein quelloffenes *Workflow Management System (WfMS)* von jBoss[3], geeignet für die Modellierung und Ausführung von Geschäftsprozessen. DESMO-J, aktiv entwickelt von der Universität Hamburg, ist ein generisches Framework für DES mit Java, das aus *Discrete-Event Simulation and Modeling (DESMO)* hervorging. Rücker formulierte folgende Anforderungen an ein DES-System [Rüc08]. In erster Linie, benötigt es ein Modell für den aktuellen Zustand. Eine Simulationsuhr, die Liste von Events, sowie ein Generator für Zufallszahlen sind erforderlich. Weiterhin benötigt wird neben statistischen Zählern und Datensammlern, eine zentrale Steuerung für die Abarbeitung der Events zum jeweilig richtigen Zeitpunkt. Für die Simulation des zeitlichen Aspekts der Bearbeitung eines Prozessschrittes, oder einer manuellen Tätigkeit, werden statistische Verteilungsfunktionen hinzugezogen. Die Statistik stellt außerdem Methoden zur Erkennung sogenannter *Warm-Up* Zeiten, innerhalb derer die Simulation keine sinnvollen Ergebnisse liefern kann, weil beispielsweise Warteschlangen noch leer sind. Auch die für eine bestimmte Genauigkeit des Ergebnisses notwendige Anzahl von Simulationswiederholungen,

[1] www.jbpm.org
[2] www.desmoj.de
[3] www.jboss.org

wird mittels statistischer Methoden erhoben. In Rückers Arbeit werden die etwaigen Service-Aufrufe entweder abgeschaltet, oder durch eine Attrappe (*Mock*) ersetzt. Pools von zur Verfügung stehenden Ressourcen werden mit Warteschlangen für Aufrufe von Prozessen versehen. Die *Prozess-Engine (PE)* von jBPM übernimmt die Ausführung der Simulationen und übergibt Events an die Simulations-Engine von DESMO-J. Voraussetzung dafür ist die Synchronisation beider Uhren. Während der Ausführung eines Prozesses zeichnet die Simulations-Engine Kennzahlen auf. Die Prozess-Durchlaufzeit, wie auch die Wartezeit für Prozesse bei der Anfrage von Ressourcen, wird zur weiteren Verarbeitung vorbereite.

Die Interaktion zwischen den beiden Engines verläuft wie folgt. Vor der Verarbeitung eines Start-Events, wird die Modell-Zeit der Simulationsuhr an die PE übertragen. Diese nicht-reale Zeit führt zu einem realistischen Simulations-Log. Das Event, ein *Plain old Java-Object (POJO)*, wird ausgeführt. Nach dem Start einer neuen Instanz des Prozesses, wird dieser bis zum ersten blockierenden Zustand durchlaufen. Darauf wird ein neues Event an die Simulations-Engine übergeben, welches später auch für die Beendigung des aktuellen Zustands verantwortlich ist. Es entspricht einem externen Ereignis, wie beispielsweise der Ankunft eines Pakets. Der Zeitpunkt, zu dem dieses Ereignis eintreten soll, wird durch eine vorkonfigurierte statistische Verteilungsfunktion festgelegt.

Um einen Prozess mit dem System von Rücker simulieren zu können, wird eine Prozessmodellierung mit BPMN, WS-BPEL, oder einer anderen mit jBPM kompatiblen Beschreibungssprache vorausgesetzt. Weiterhin muss eine Simulationskonfiguration in der XML angelegt werden, welche die Verteilungsfunktionen für Laufzeiten von Prozessschritten festlegt und verschiedene Szenarien enthalten kann. Einige der benötigten Parameter können auch aus den Logs vorangegangener Prozessausführungen des Produktivsystems ermittelt werden. Die Szenarien beschreiben Konfigurationsvariationen von Ressourcen-Pools und Kapazitäten der Warteschlangen. Prozessvariablen werden durch Datenquellen definiert, die einem Stream von Datenobjekten entsprechen. Außerdem können diese durch Datenfilter, nach dem *Pipes & Filters* Entwurfmuster, während der Simulation modifiziert werden. Ebenfalls Teil der Simulationskonfiguration, ist das Behandeln von Service-Aufrufen durch den Prozess.

Rückers System für BPS unterstützt bei der Überprüfung von Hypothesen, führt die Optimierung des Prozesses jedoch nicht automatisch durch. Das Wissen um statistische Verteilungen von Parametern der Prozessschritte ist explizit zu formulieren. Vorangegangene Ausführungszeiten können durch Process-Mining erfasst und die Verteilung approximativ festgelegt werden. Eine Erweiterung der PE ist notwendig.

Distributed Simulation Optimization (DISMO) ist ein Framework für die automatische Instanziierung und Ausführung von Experimenten. Ziel ist die Optimierung von Modellparametern durch parallele, verteilte Simulation. DISMO verbindet DES und heuristische Optimierungsverfahren im Kontext verteilter Systeme [GP01]. Gehlsen et al. beschreiben das Simulationsmodell als Black-Box Funktion über den Ein- und Ausgabevektoren. Ein genetischer Algorithmus (GA) bildet die Grundlage zur Optimierung der durch DESMO-J ausgeführten Simulation. Dieser heuristische Algorithmus führt eine direkte Suche über dem Lösungsraum eines Problems aus und bedient sich bei der Optimierung von Parametern am Prinzip der biologischen Evolution. Das Populationskonzept des GA ermöglicht die Parallelisierung der Simulation, da die Individuen, beziehungsweise Lösungskandidaten des Optimierungsproblems, voneinander unabhängig sind. Die Koordination der Verteilung unterschiedlicher Experimente wird durch *Remote Method Invocation (RMI)* bewerkstelligt. Bis ein definiertes Abbruchkriterium erfüllt ist, werden Simulationen zyklisch wiederholt. Danach bewertet eine Fitnessfunktion das Ergebnis auf Tauglichkeit im Bezug auf den Eingabevektor. Eine neue Population wird generiert und die Optimierung fortgeführt. Eine Integration von DISMO in BPS ermöglicht die automatische, parallelisierte Prozessoptimierung.

4 Anforderungen

Auf den im vorangegangenen Kapitel vorgestellten Konzepten aufbauend, werden in diesem die Anforderungen einer Integration der MAPE-K-Rückkopplungsschleife in Prozesse innerhalb CPS erhoben. Sowohl bestehende Forschungsarbeiten, als auch die in dieser Arbeit vorgestellten Konzepte für solch eine Integration, werden anhand dieser evaluiert. Broy et al. und Gurgen et al. postulierten Anforderungen für CPS beziehungsweise autonome Systeme, welche die Grundlage der in dieser Arbeit vorgestellten bilden [BCG12; Gur+13]. Eine Tabelle visualisiert für einige Abschnitte die Korrelation des Forschungsstands mit den Anforderungsaspekten durch *erfüllt* (+), *unerfüllt* (-) und *beschränkt* ((+)).

4.1 Infrastruktur

Service-Aufrufe in Prozessen, die Aggregation von Sensordaten und das Manipulieren von Aktuatoren, muss zuverlässig und erwartungskonform ermöglicht werden. Grenzen eines Netzes zwischen diesen Entitäten sollen dynamisch auf veränderliche Anforderungen und Umgebungsparameter reagieren können, was mit der Flexibilisierung von Ressourcen und deren Kapazität, beziehungsweise Elastizität, einher geht [BCG12]. Der damit notwendige Austausch, oder das Fehlen von System-Komponenten muss realisiert beziehungsweise angemessen kompensiert werden. Komposition, Integration und dezentrale Kontrolle innerhalb eines Systems von Systemen erfordert die Erkennung der Situation, aktive Suche nach passenden Komponenten und das Einbetten fehlender Teile, wie Services, Daten und Funktionen [BCG12]. Eine adäquate Infrastruktur ermöglicht Modularität, Anpass- und Erweiterbarkeit im Bezug auf Soft- und Hardware [Gur+13].

R01 Modularität, Fehlertoleranz und Elastizität zeichnen jene Infrastruktur aus, innerhalb derer CPS-Prozesse ausgeführt werden und Kontroll- beziehungsweise Überwachungsmechanismen situations- und verhaltensabhängig agieren.

Tabelle 4.1: Forschungsstand und Infrastruktur

	[Sch+13]	[MRD08]	[BG05]	[JMM12]	[KK12]	[Ope92]	[OAS09]	[Rüc08]
modular	+	+	+	+	+	+	+	+
fehlertol.	+	+	-	-	-	-	-	-
elastisch	+	+	-	-	-	-	-	-

Bei ViePEP und VieDAME (vgl. Abschnitt 3.1) bilden infrastrukturelle Aspekte den Schwerpunkt [Sch+13; MRD08]. Durch Services als zentralen Bestandteil, ist neben dem Aufbau dieser auch der des Systems von Baresi et al. modular [BG05]. Da die Auslastung einzelner Komponenten und die Bewertung von QoS-Attributen in die Antizipation des Ressourcenbedarfs eingeht, können die Systeme von Schulte et al. beziehungsweise Moser et al. Fehler tolerieren und agieren elastisch. Das Modularitätskriterium ist durch Janiesch et al. (vgl. Abschnitt 3.4) erfüllt, da das Event-basierte Konzept mit beliebigen EPAs betrieben und eine Aufteilung nach Zuständigkeiten in Event-Producer, -Consumer und -Processor vorgenommen wird [JMM12]. Auch Kötter et al. haben in der aPro (vgl. Abschnitt 3.5) eine modulare, ereignisgesteuerte Infrastruktur mit unabhängigen Monitoring-Services [KK13]. Neben der XA für verteilte Transaktionen, ist auch WS-TX (vlg. Abschnitt 3.6) und das Konzept von Rücker (vlg. Abschnitt 3.7) modular strukturiert [Ope92; OAS09; Rüc08].

4.2 Interoperabilität

Verteilte Systeme erfordern einen hohen Aufwand an Kommunikation und Synchronisation. Durch die Heterogenität von CPS sind standardisierte Komponenten-Schnittstellen für die Interoperabilität unerlässlich [Gur+13]. Auch die zielgerichtete Adaption von Interaktion, Koordination, sowie Kontrolle von und mit anderen Systemen und Services wird erst durch Standards ermöglicht [BCG12]. Weiterhin ist der von Seiger et al. diskutierte Aspekt der Dynamik ein zu berücksichtigender [Sei+13]. Nach deren Prozessmodell (vgl. Abschnitt 2.3) soll die Model-

lierung von Service-Aufrufen und Prozessschritten unabhängig von der konkreten Komponente sein, wodurch der Einsatz von Schnittstellenstandards Voraussetzung wird.

R02 Die Kommunikation zwischen der Prozessinstanz und den Komponenten der Rückkopplungsschleife innerhalb des CPS, erfolgt über standardisierte Schnittstellen.

Durch die aspektorientierte Überprüfung des SOAP-Nachrichtenaustauschs zwischen verschiedenen Services, ist die Interoperabilität bei dem Projekt VieDAME (vgl. Abschnitt 3.1) gewährleistet [MRD08]. Das Konzept von Janiesch et al.(vgl. Abschnitt 3.4) nutzt den standardisierten JMS[1] für die Kommunikation zwischen den Komponenten des Systems [JMM12]. XA und WS-TX (vgl. Abschnitt 3.6) sind selbst Standards und ermöglichen somit Interoperabilität [Ope92; OAS09].

4.3 Selbstwahrnehmung

Für dynamische Infrastrukturen und reibungslose Kommunikation muss der eigene Zustand einer Komponente hinreichend bekannt sein. Da in CPS kein zentrales Kontrollorgan existiert, sind Services und Geräte selbst in der Verantwortung Leistungsschwankungen und Ausfälle auszugleichen um die bestmögliche QoS bieten zu können. Ein Prozess kann sich dadurch für den Service höchster Güte entscheiden [BCG12]. Management- und Performanz-Daten besitzen damit, zumindest für die einzelne Komponente, die gleich Priorität wie Sensordaten für die Wahrnehmung der Realität [Gur+13]. So sollten Softwaremodule, von Anwendungslogik getrennt, eine selbstreflexive Form der Überwachung implementieren, um Situation, beziehungsweise den aktuellen Kontext des Systems, Zustand und Handlungsoptionen in etwaige Anpassungsentscheidungen einfließen lassen zu können [Gur+13; BCG12].

R03 Eigene Situation, Zustand und Handlungsoptionen werden vom System wahrgenommen und in die Entscheidungen einer Rückkopplungsschleife integriert.

[1] Java Message Service

Tabelle 4.2: Forschungsstand und Selbstwahrnehmung

	[Sch+13]	[MRD08]	[BG05]
Situation	-	+	+
Zustand	+	-	-
Handlungsoptionen	+	+	+

Mit der Überwachung der Antwortlatenzen und Verfügbarkeit einzelner Komponenten des Gesamtsystems, sind die Konzepte von Moser et al. und Baresi et al. im Stande, die Situation zu erfassen [MRD08; BG05]. Der Zustand eines Services wird innerhalb von ViePEP (vgl. Abschnitt 3.1) durch seine Auslastung repräsentiert [Sch+13]. Die auf den Austausch von Services fokussierten Handlungsoptionen bei Schulte et al. und Moser et al., sind eine Variante der Rückkopplung [Sch+13; MRD08]. Auch Baresi et al. (vgl. Abschnitt 3.2) implementierten Prozess-Resonanz durch den Monitoring-Manager, ohne Einfluss auf die folgende Aktion [BG05].

4.4 Daten und Operationen

4.4.1 Aggregation

Das Erfassen des physikalischen Kontextes, beginnend mit der Aggregation von Sensordaten, ist ein zentraler Aspekt rückgekoppelter Prozesse in CPS. Nur die Wahrnehmung der Realität erlaubt eine Evaluation von Zielen und das Erkennen von Erfolg und Scheitern eines Prozesses mit realen Effekten. Jedoch erschweren heterogene Sensor-Landschaften mit Messfehlern und Ausfällen die Zuverlässigkeit dieser. Eine Verknüpfung unterschiedlicher Quellen von Information und das Verrechnen der Werte erlaubt den Ausgleich, das Präzisieren des Modells der physikalischen Umgebung und die Erkennung der Situation [BCG12]. Weiterhin produziert eine große Zahl von Geräten eine erhebliche Datenmenge, die in Echtzeit erfasst werden muss. Somit sind Parallelität und geeignete Filter bei der Verarbeitung eines solchen Stroms von Information unerlässlich.

R04 Parallelisierte Echtzeiterfassung physikalischer Daten, sowie deren Fusion, erlaubt die Wahrnehmung von Kontext und Situation.

Tabelle 4.3: Forschungsstand und Aggregation

	[Sch+13]	[RA08]	[MC10]	[DAV12]	[JMM12]	[KK12]
parallel	-	-	-	-	+	+
echtzeitlich	-	-	-	-	+	+
von Rohdaten	+	(+)	(+)	(+)	+	+
durch Datenfusion	-	-	-	-	+	+

Das parallele Erfassen physikalischer Parameter, wie Prozessor- und Speicherauslastung, bildet den Ursprung der Adaptivität von ViePEP [Sch+13]. Bei Rozinat et al., Muñoz-Gama et al. und De Leoni et al. ist die Aggregation von Daten auf Log-Dateien beschränkt, welche Indirektionen zu physikalischen Daten beinhalten können [RA08; MC10; DAV12]. Parallelität und echtzeitliches, direktes Erfassen physikalischer Daten, ist sowohl mit dem System von Janiesch et al. (vgl. Abschnitt 3.4), als auch mit jenem von Kötter et al. (vgl. Abschnitt 3.5) durch CEP-Streaming und daran gekoppelte Events möglich. Die Fusion der Daten wird bei Janiesch et al. durch Event-Producer und bei Kötter et al. mit Metriken und KPIs abstrahiert [JMM12; KK13].

4.4.2 Modell

Erfasste Daten müssen für Echtzeit- und retrospektiv-Analysen in einer definierten Form persistiert werden. Ein generisches, semantisches und erweiterbares Modell für Metadaten, Laufzeit- und Kontroll- und Umgebungsinformationen wird benötigt [Gur+13]. Nach Broy et al. repräsentiert dieses neben dem Kontext auch die An-

wendungsdomäne, hier mit Informationen zu Prozessen und deren Zielen [BCG12]. Weiterhin wird der Lebenszyklus jedes Datums abgelegt [Gur+13]. In Summe ermöglichen diese Daten einem autonomen System die Interpretation und Vorhersage von Fehlern, Risiken und Hürden [BCG12]. Keine der vorgestellten Arbeiten besitzt ein derartiges Modell.

R05 Ein generisches, semantisches und erweiterbares Modell dient der strukturierten Persistenz von Kontext-, Prozess- und Metadaten.

4.4.3 Konsistenz

Nicht nur die Veränderung der Modell-Daten, auch Operationen in Form von Befehlen an Aktuatoren sollen Konsistenzkriterien entsprechen. Nach Gurgen et al. muss die Beeinflussung von Kontextcharakteristika und damit die intentionale Veränderung realitätsbezogener Parameter transaktionale Eigenschaften wie Atomizität, Isolation und Dauerhaftigkeit besitzen. Weiterhin existiert ein veränderliches Maß, das dem Nutzer erlaubt den Kompromiss zwischen Ressourcenverbrauch und Konsistenzgrad zu optimieren [Gur+13].

R06 Die Manipulation des realen Kontextes durch Rückkopplungsschleife und Prozess unterliegt variablen transaktionalen Prinzipien zur Wahrung operationaler Konsistenz.

Tabelle 4.4: Forschungsstand und Konsistenz

	[GS87]	[Zha+94]	[Ope92]	[OAS09]
transaktionale Kontextmanipulation	+	+	+	+
variabler Transaktionalitätsgrad	-	+	-	+

Das Sagas-Modell von Garcia-Molina et al. (vgl. Abschnitt 3.6) bildet den Ursprung vieler Transaktionskonzepte, bietet Atomizität durch Kompensation in Prozessen und somit ein zentrales Konsistenzprinzip [GS87]. Zhang et al. erweiterten dieses um Semiatomizität und ermöglichten einen variablen Transaktionsgrad durch die vorgeschlagenen Prozessschritt-Typen [Zha+94]. Mit XA und WS-TX kann transaktionale Kontextmanipulation durch zusätzliche Indirektion, zum Beispiel einen

Service, durchgesetzt werden [Ope92; OAS09]. Variabilität bietet WS-TX bezüglich der verschiedenen Protokollvarianten.

4.4.4 Analyse

Für Identifikation von Handlungsoptionen und Evaluation des Prozesserfolgs ist eine Analyse der Kontext- und Anwendungsdaten notwendig, die aufgrund der kontinuierlichen Aggregation nahezu in Echtzeit erfolgen muss [Gur+13; BCG12]. Zur Minimierung des Kommunikationsaufwands zwischen Prozess, Rückkopplungsschleife und CPS-Komponenten generell, sollte eine gerätespezifische, lokale Analyse bevorzugt genutzt werden [Gur+13]. Somit ist es einem CPS möglich, die aktuelle Situation in Abhängigkeit von Zielen und Fortschritt von Aufgaben, beziehungsweise Prozessen zu erfassen [BCG12].

R07 Die bevorzugt gerätespezifische Analyse der Kontext- und Anwendungsdaten geschieht nahezu in Echtzeit und ermöglicht die Evaluation von Situation, Handlungsoptionen, Prozesszielen und -erfolg.

Tabelle 4.5: Forschungsstand und Analyse (EZ: echtzeitlich, EHO: Evaluation d. Handlungsoptionen, PZ/PE: Prozessziele/-erfolg)

	[Sch+13]	[MRD08]	[BG05]	[JMM12]	[KK12]	[Ope92]	[OAS09]	[Rüc08]
EZ	+	+	+	+	+	-	-	-
EHO	+	+	-	-	-	-	-	-
PZ/PE	-	-	-	-	+	(+)	(+)	+

QoS-Attribute wie die Antwortzeit werden mit ViePEP und VieDAME (vgl. Abschnitt 3.1) nahezu in Echtzeit analysiert. Durch Reasoning und modulare Service-Ersetzungsstrategien evaluieren Schulte et al., beziehungsweise Moser et al. die für das System relevanten Handlungsoptionen [Sch+13; MRD08]. Bei Baresi et al. (vgl. Abschnitt 3.2) wird der Aspekt echtzeitlicher Analyse durch das Proxy-Konzept erreicht [BG05]. In der Arbeit von Janiesch et al. (vgl. Abschnitt 3.4) wird das

BAM-Echtzeitkonzept erweitert [JMM12]. Kötter et al. (vgl. Abschnitt 3.5) ermöglichen die Analyse in Echtzeit durch die Verwendung von CEP. Ebenfalls in dieser Arbeit erfolgt die Evaluation von Prozesszielen und dessen Erfolg durch KPIs und deren Fusion [KK12]. Durch 2PC ist es der XA (vgl. Abschnitt 3.6) indirekt möglich Prozessziele und -erfolg zu ermitteln [Ope92]. Ein Commit kann auch Prozesserfolgsindikator für Systeme mit WS-TX sein [OAS09]. Bei Rücker bietet die Simulation eines Prozesses (vlg. Abschnitt 3.7) diese Evaluation noch vor dem produktiven Ausführen desselben [Rüc08]. Die Anforderung der bevorzugt gerätespezifischen, beziehungsweise lokalen Analyse, sowie die der Situation wird von keiner der vorgestellten Arbeiten erfüllt.

4.5 Handlungsselektion

Trotz der erheblichen Menge an Information innerhalb eines CPS müssen Entscheidungen über weiteres Vorgehen meist auf Basis unsicheren, teils widersprüchlichen Wissens getroffen werden [BCG12]. Ereignisregeln, Zielfunktionen oder Vorhersagemodelle, wie Entscheidungsbäume werden populiert und erlauben zielführendes Handeln, wobei Konflikte zwischen verschiedenen Regeln durch das System zu lösen sind [Gur+13]. Die konkrete Bewertung von Zielen und Schritten einer Komponente oder eines Prozesses aus der Analysephase, sowie Kosten und Risiken der möglichen Aktionen wird in die Selektion weiterer Handlungen einbezogen [BCG12].

R08 Die Bewertung von Zielen und Situation befähigt ein CPS zur Selektion weiterer Aktionen, wobei Kosten und Risiken dieser, sowie Ereignisregeln, Regelkonfliktmanagement, Zielfunktionen oder Vorhersagemodelle genutzt werden.

Tabelle 4.6: Forschungsstand und Handlungsselektion

	[Sch+13]	[MRD08]	[GP01]
Kosten & Risiken	+	+	-
Regeln/Zielfunkt./Vorhersagemod.	+	+	+
Regelkonfliktmanagement	-	-	-

Sowohl Schulte et al., als auch Moser et al. (vgl. Abschnitt 3.1) integrieren Kosten und Risiken durch die QoS eines Dienstes indirekt in die Selektion der Handlung. Während Reasoning in ViePEP genutzt wird, stellt beispielsweise das Unterschreiten eines bestimmten QoS-Niveaus die zutreffende Regel für den Selektionsprozess bei VieDAME [Sch+13; MRD08]. DISMO von Gehlsen et al. (vgl. Abschnitt 3.7) nutzt GAs zur Optimierung des Eingabevektors einer Prozess-Simulation [GP01]. Das Management von Regelkonflikten wird von keiner der vorgestellten Arbeiten übernommen.

4.6 Lernen

Durch Prozess, oder Rückkopplungsschleife initiierte und abgeschlossene Aktionen führen zu Erfahrungswissen bezüglich der Qualität einer Handlungsselektion. Lernen durch fallbasiertes Schließen, erlaubt die Integration dieser zusätzlichen Informationen in künftige Analysen und Entscheidungen [Gur+13]. Das dadurch ermöglichte Erkennen und Vermeiden von Fehlern, Hürden und Risiken, sowie die Korrektur von Parametern, verhilft einem autonomen System zu inkrementeller Verbesserung [BCG12]. Mit der Veränderung von Interaktion, Prozessen oder Abläufen, kann das CPS sein Verhalten zielgerichtet und adaptiv korrigieren [BCG12].

R09 Komponenten von CPS verbessern inkrementell ihre Analyse- und Entscheidungsiterationen durch erfahrungsorientiertes Lernen um autonom zielgerichtete Adaptivität zu erreichen.

Tabelle 4.7: Forschungsstand und Lernen

	[KK12]	[Rüc08]	[GP01]
Erfahrungslernen	-	(+)	(+)
inkr. Verbesserung	-	(+)	+
autonom. Adaption	(+)	-	-

Die Integration der Ergebnisse von Prozess-Simulationen ist Teil der Arbeiten von Rücker und Gehlsen et al. (vgl. Abschnitt 3.7). Während Rücker die inkrementelle Verbesserung eines Prozesses durch DES ermöglicht, optimieren Gehlsen et al. den Eingabevektor dessen mit GAs [Rüc08; GP01]. Eingeschränkt möglich wird das Lernen in diesen Konzepte durch ein manuelles Übertragen der Ergebnisse von Simulation und Optimierung. Der Aspekt autonomer Adaption mittels Lernmechanismen wird von der vorgestellten Arbeiten nicht abgedeckt. Lediglich Kötter et al. (vgl. Abschnitt 3.5) forschen an einer Erweiterung durch ABIS [KK12].

4.7 Reaktion

Die Adaption an veränderte Situationen und Anforderungen erfordert das Ausführen konkreter Aktionen um den gewünschten Zustand, beziehungsweise ein Prozessziel zu erreichen. Durch die Modifikation von Aktuator-, Netzwerk- oder Performanzparametern, sowie die Installation und Deinstallation von Softwaremodulen, wird ein CPS dazu befähigt [Gur+13]. Der Zugriff auf die dafür benötigten Ressourcen muss kontrolliert und verwaltet werden [BCG12].

R10 Der Prozess, die Rückkopplungsschleife und Komponenten des CPS generell, kompensieren eine Abweichung vom gewünschten Zustand durch Aktuatoren und das Anpassen von Parametern bezüglich Netzwerk und Performanz, sowie das Management benötigter Softwaremodule, wobei der Zugriff auf diese Ressourcen kontrolliert erfolgt.

Tabelle 4.8: Forschungsstand und Reaktion

	[Sch+13]	[BG05]	[JMM12]	[KK12]	[Ope92]	[OAS09]	[GP01]
Aktuatoren	-	-	+	(+)	-	-	-
Parameter	-	+	-	-	-	-	+
Softwaremodulmgt.	+	-	-	-	-	-	-

Tabelle 4.8: Forschungsstand und Reaktion

	[Sch+13]	[BG05]	[JMM12]	[KK12]	[Ope92]	[OAS09]	[GP01]
Ressourcen-zugriff	-	-	-	-	+	+	-

Das Starten und Stoppen von Services, als Management von Softwaremodulen, zur Reaktion auf den aktuellen Ressourcenverbrauch ist im Konzept von ViePEP (vgl. Abschnitt 3.1) enthalten [Sch+13]. Mit indirektem Einfluss auf die Performanz, ist der Grad der Überwachung bei Baresi et al. (vgl. Abschnitt 3.2) ein manuell veränderlicher Parameter [BG05]. Auch das Simulationskonzept von Gehlsen et al. (vgl. Abschnitt 3.7) ermöglicht die Anpassung und Optimierung von Parametern bezüglich der Performanz eines Prozesses [GP01]. Janiesch et al. ermöglichen das Einbinden von Aktuatoren als Event-Consumer (vgl. Abschnitt 3.4) und Kötter et al. Visualisieren die Ergebnisse (vgl. Abschnitt 3.5) eines mit Zielen angereicherten Prozessmodells in einem Dashboard [JMM12; KK13]. Mit diesem ist bei Kötter et al. die Anbindung an eine konsumierende Komponente zumindest vorbereitet. Kontrollierter Ressourcenzugriff wird von XA und WS-TX (vgl. Abschnitt 3.6) durch den Resource-Manager, beziehungsweise das WS-Coordination-Protokoll bereitgestellt [Ope92; OAS09].

4.8 Sicherheit

Der kontrollierte Zugriff auf Ressourcen ist ein Aspekt der Sicherheit in CPS [BCG12]. Durch die Arbeit mit sensiblen, teils persönlichen Daten und das Beeinflussen der physikalischen Umgebung, muss neben der Betriebssicherheit (*Safety*) auch die Angriffssicherheit (*Security*)[1] in den Entwurf eines solchen einbezogen werden [Gur+13]. Durchgesetzt werden müssen außerdem Validation und Verifikation jedweder Daten, sowie der Schutz von Privatsphären involvierter Personen [BCG12].

[1] de.wikipedia.org/wiki/Sicherheit

R11 Cyber-physikalische Umgebungen sind bestmöglich vor böswilligen Zugriffen geschützt und bieten dem Nutzer genügend Sicherheit im Bezug auf seine Privatsphäre und persönliche Daten. Die Verifikation und Validation von Daten innerhalb des Systems wird vorausgesetzt.

Die Notwendigkeit der Sicherheit in CPS wird in vielen Arbeiten als essentiell beschrieben. Dennoch ist eine Konkretisierung, durch den Umfang und die Vielfältig der Problemaspekte, in dieser frühen Phase der Forschung kaum möglich.

Nach der Betrachtung bestehender Forschungsarbeiten und dem Aufstellen von Anforderungen, findet sich kein ganzheitliches Konzept für die Rückkopplung bei Prozessen in CPS. Jede der Arbeiten löst Teilaspekte des Ausgangsproblems, die im nächsten Kapitel in ein neues Konzept integriert werden. Sicherheit wird aufgrund des Umfangs nicht thematisiert.

5 Konzeption

Mit den in Abschnitt 4 erhobenen Anforderungen und Erkenntnissen bestehender Forschungsarbeiten aus Abschnitt 3, wird in diesem Kapitel ein Konzept zur Integration von MAPE-K in Prozesse cyber-physischer Systeme vorgestellt. Neben der konsistenten Abbildung von Realität und deren virtuellem Modell, werden die Phasen der Rückkopplungsschleife in die Prozessmodellierung und -ausführung integriert. Weiterhin sind Transaktionalität und Simulation von Prozessen Gegenstand des folgenden Entwurfs.

5.1 Komponenten von MAPE-K

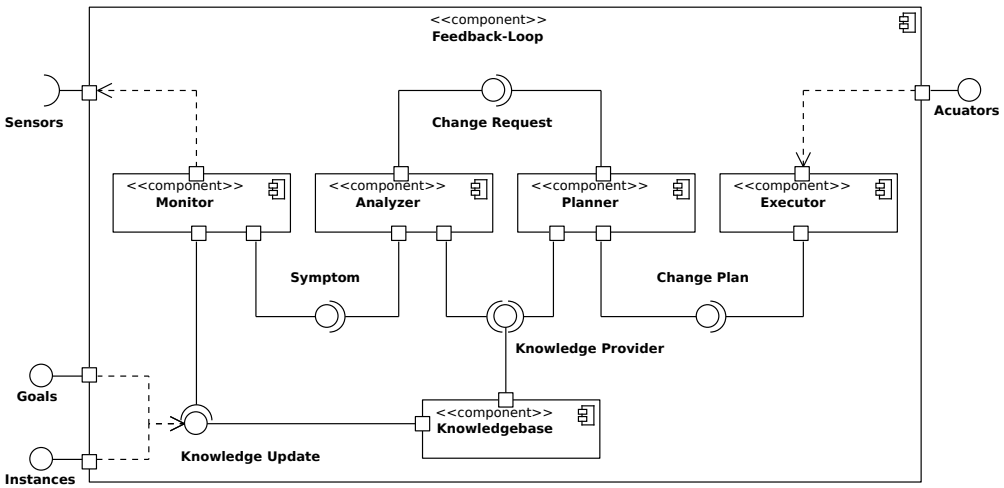

Abbildung 5.1: Komponenten der Kontrollschleife für Prozesse

Nicht zuletzt durch Anforderung R01 ist die modulare Strukturierung des zu entwerfenden Systems essentiell. MAPE-K und die Prozess-Engine (PE) sind aufgrund ihrer unterschiedlichen Zuständigkeiten voneinander zu trennen. Die Verbindung dieser beiden Komponenten wird in Abschnitt 5.2 detailliert erläutert. Abbildung 5.1 zeigt den in diesem Konzept vorgestellten Aufbau der Kontrollschleife, wobei die einzelnen Phasen eigenständigen Modulen entsprechen.

5.1.1 Knowledge

Prozesse in CPS sind Teil eines realen Kontexts, der konsistent abgebildet werden muss. Neben Symptomen, Regeln, Change-Requests und weiteren für MAPE-K notwendigen Informationen (vlg. Abschnitt 2.4), ist ein virtuelles Abbild der Realität Baustein des für die Rückkopplung benötigten Wissens. In Übereinstimmung mit R05, ist das zugrundeliegende Datenbanksystem generisch, erweiterbar und enthält semantische Informationen. Eine relationale Datenstruktur wäre zu inflexibel bei der Abbildung eines heterogenen Kontexts, da die räumliche Organisation der realen Umgebung durch komplexe Verbindungen geprägt ist. Wenn beispielsweise das zweite Fenster im Wohnzimmer des Erdgeschosses einen Sensor für den Zustand des Schließmechanismus besitzt, ist die Abfrage des aktuellen Wertes nur über die Verkettung (Join) mehrerer Tabellen möglich. Für die Aktualisierung des Zustands muss dieser Pfad ebenfalls hergestellt werden. Aufgrund der Komplexität und der damit einhergehenden Laufzeitverlängerung einer solchen Operation, ist die Echtzeitaktualisierung des Kontextmodells nicht möglich. Graph-basierte Modelle besitzen kein festes Schema (Generizität) und ermöglichen Zuordnungen über Tabellen und Zeilen hinaus. Weiterhin erlauben sie das Speichern typisierter Schlüssel-Wert-Paare (Attribute) auf Basis der Knoten und Relationen, wodurch nicht jede Kontextinformation eine eigene Relation benötigt.

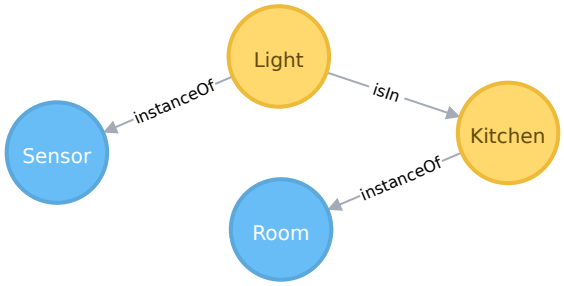

Abbildung 5.2: Modell für die Beleuchtung in der Küche

Um das Ergebnis eines Prozesses, oder seiner Teile messen zu können, müssen involvierte Geräte lokalisierbar sein. Eine direkte Adressierung, zum Beispiel über eine ID, ist in diesem Zusammenhang kaum möglich, da der verwendete Sensor/Aktuator während der Modellierung des Kontextmodells nicht konkret benannt werden soll. Diese Forderung nach Generizität, ist neben R05 (vgl. Abschnitt 4.4.2) auch in den Anforderungen an Service-Aufrufe von Seiger et al. (vgl. Abschnitt 2.3) verankert [Sei+13]. In der Domäne des Smart Home (SH), befinden sich Sensoren und Aktuatoren in Räumen oder an Orten, wodurch eine indirekte Referenz, beziehungsweise ein Pfad, existiert. Der Verbund zwischen einem Raum und den darin befindlichen Sensoren lässt sich, wie in Abbildung Abbildung 5.2, durch einen Graphen beschreiben. Der Knoten `Light` repräsentiert die Instanz eines Lichtsen-

sors, was durch die Relation `instanceOf` beschrieben wird. Außerdem befindet sich dieser Sensor in der Küche (`Kitchen`), dargestellt durch die Verbindung `isIn`. Der Helligkeitswert selbst, ist entweder ein Attribut von `Light` oder ein eigenständiger Knoten mit dem konkreten Wert. Durch dieses Modell lassen sich Ziele eines Schrittes, oder die eines Prozesses selbst festlegen (z. B. *Tageslicht in der Küche*). Diese Beschreibung entspricht einem Pfad im Graphen, der mit einer Assertion (Zusicherung) verbunden werden kann. Listing 5.1 ist ein mit Cypher[1] formuliertes Beispiel für eine Assertion auf diesem Pfad, wobei hier der Wert für Tageslicht 800 Lux entspricht.

Listing 5.1: Kontextpfad und Assertion von Tageslicht in der Küche

```
1 MATCH (kitchen)-[:instanceOf]->(room)
2 MATCH (light)-[:instanceOf]->(sensor)
3 MATCH (light)-[:isIn]->(kitchen)
4 RETURN light.value > 800
```

Anhand von Abfragen, wie der in Listing 5.1, wird in der Analyse-Phase von MAPE-K der Erfolg eines Prozesses ermittelt. Die Einschätzung des Prozesserfolgs und das Definieren von Prozesszielen, wurde bereits von Kötter et al. untersucht [KK12]. Durch den Fokus auf das Prozessmonitoring, ließen sich mit der ProGoalML die Metriken und KPIs festlegen, nicht aber der Pfad in einem Kontextgraphen und damit verbundene Adressierung des Messpunkts (vgl. Abschnitt 3.5). Diese Konzept teilt Ziele in Goals und Objectives (vgl. Abschnitt 2.4) auf, die über `Knowledge Update` vom Prozess bereitgestellt werden. Policies werden mit der Erzeugung des Modells festgelegt, da sie sich auf die Struktur des spezifischen Kontexts beziehen. An den Prozess gebundene Policies bleiben in dieser Arbeit unbetrachtet, da sie als Teil der Prozessdefinition und nicht der Kontrollschleife angesehen werden. Das Goal eines Prozesses, hält Relationen zu Objectives (`hasObjective`), die den Kontextpfad, beziehungsweise den Pfad des Messpunktes (`testNode`), und eine zeitliche Begrenzung als Attribut beinhalten. Eine Prozessinstanz P ist ein Knoten im Graphen, verbunden mit dem konkreten Kontext C und dem Goal G wie in dem Beispiel aus Abbildung 5.3.

Neben Graphen im Allgemeinen, ist das *Resource Description Framework (RDF)*[2], ein Standard für das semantische Web, ebenfalls für derartige Abbildung geeignet. In diesem werden Relationen in einem Tripel aus Subjekt, Prädikat und Objekt definiert. Zeile eins in Listing 5.1 lässt sich in ein solches überführen, wobei `kitchen` das Subjekt, `instanceOf` das Prädikat und `room` das Objekt darstellt. Für die Modellierung eines solchen SH-Kontextmodells eignet sich DogOnt von Corno et al. [CB08]. Diese Ontologie erlaubt die Definition von Räumen und darin

[1] neo4j.com/docs/stable/cypher-introduction.html
[2] www.w3.org/RDF

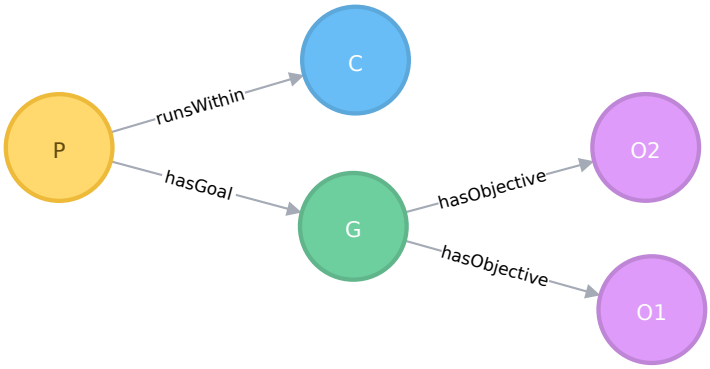

Abbildung 5.3: Modell für einen Prozesses und dessen Verbindungen

befindlichen Geräten, sowie eine Abbildung der Wechselwirkungen zwischen den verschiedenen Komponenten des CPS. Während die RDF einen durch Tripel beschränkten Graphen beschreiben, sind allgemeine Graphen frei bezüglich Richtung und Anzahl von Kanten, wodurch die Definition des Kontextpfads vereinfacht wird (vgl. Abbildung 5.2). Weder Reasoning, noch die Begrenzung des Graphen werden in diesem Konzept benötigt, wodurch das RDF für die Wissensbasis (WB) nicht verwendet wird. Die Schnittstelle `Knowledge Update` und `Knowledge Provider` (Abbildung 5.1) dienen dem Aktualisieren des Kontexts, dem Erfassen der Ziele des Prozesses und dem Bereitstellen von Informationen für die folgenden Komponenten von MAPE-K. Das Modell dieser Komponente muss im Vorhinein aufgebaut werden um eine direkte Laufzeit-Integration zu gewährleisten.

5.1.2 Monitor

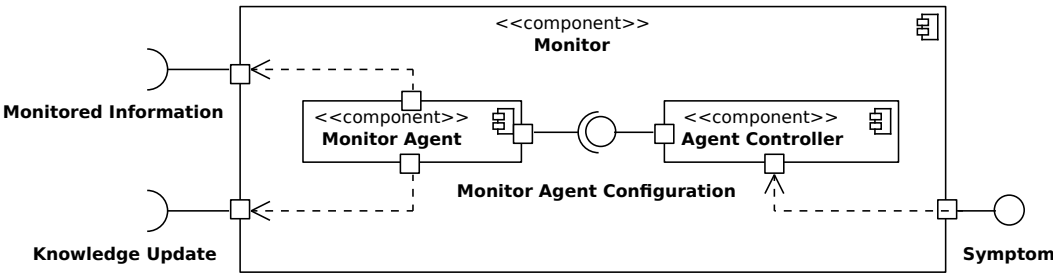

Abbildung 5.4: Detailansicht der Monitor-Komponente mit einem Agent

Die Überwachung des Kontexts eines Prozesses in CPS wird durch die Aggregation von Events unterschiedlicher Subsysteme bestimmt (dritter Schritt in Abbildung 5.5). Das System ist im Stande mehrere Kontexte zeitgleich zu überwachen, wodurch eine `Monitor`-Instanz genau einem Kontextmodell zugeordnet wird. Ein

`Monitor Agent`, wie in Abbildung 5.4, bildet die Schnittstelle zu einer von vielen heterogenen Quellen der Information und arbeitet parallel zu den anderen (vgl. Abschnitt 4.4.1). Abbildung 5.5 zeigt detailliert die Interaktion des `Agent Controller` mit den `Monitor Agents` und der `Knowledgebase`. Die Integration einer CEP-Engine, einer IoT-/Sensor-Middleware, oder das Abfragen der Prozess- und Geräte-Logs, besitzt je einen eigenen Agent. Auch der in der Arbeit von Janiesch et al. beschriebenen Event-Consumer (vgl. Abschnitt 3.4) kann als ein `Monitor Agent` verstanden werden. Die von ViePEP und VieDAME gesammelten Daten zu Auslastung von Speicher, CPU und Antwortzeit von Services (vgl. Abschnitt 3.1) können in diese MAPE-K-Phase durch Agents einbezogen werden. Es obliegt den Agents, fehlerbehaftete Events oder Ungenauigkeiten der Werte im Vorhinein zu verarbeiten und zu homogenisieren, um die durch Quellenheterogenität entstandenen Differenzen auszugleichen (vgl. Abschnitt 4.4.1, vierter Schritt in Abbildung 5.5). Für die Aktualisierungen des Kontextmodells, muss diesen Komponenten die Struktur dessen bekannt sein.

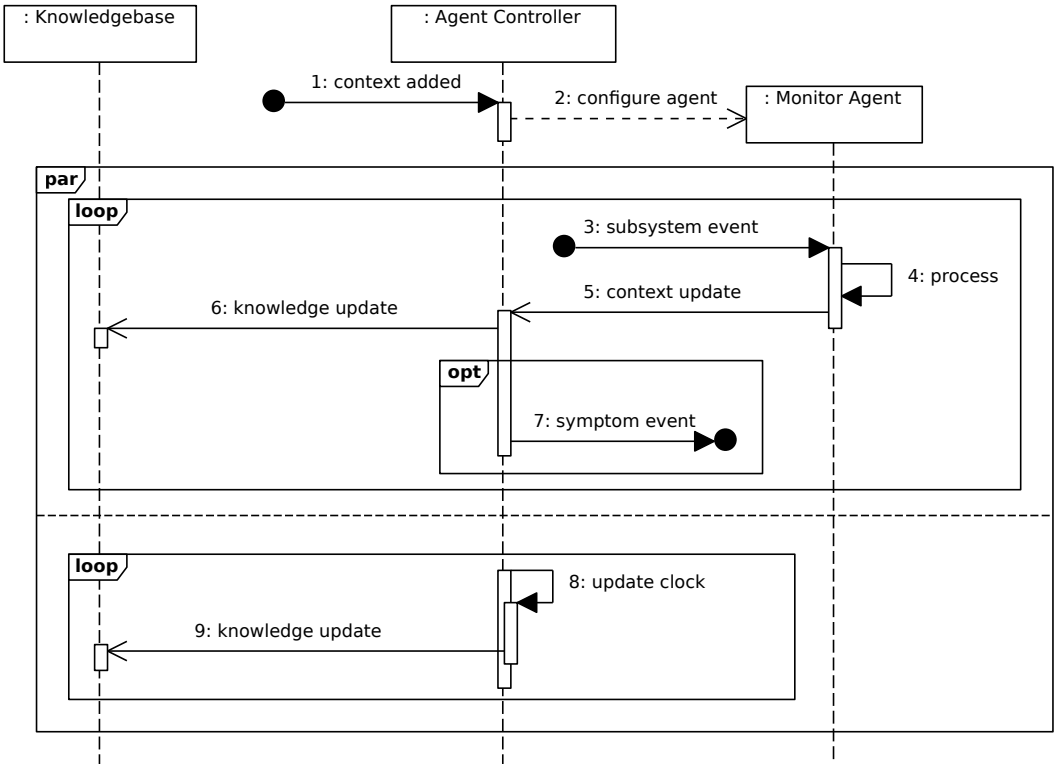

Abbildung 5.5: Monitoring in einer MAPE-K-Schleife

Der übergeordnete `Agent Controller` aggregiert die Ergebnisse der einzelnen Agents und ist zuständig für das Aktualisieren der Kontextmodellzeit, ähnlich der BPS-Uhr (vgl. Abschnitt 3.7, achter Schritt in Abbildung 5.5). Diese Zeit wird für temporale

Assertions in den Objectives benötigt und ermöglicht beispielsweise, die Zeit für das Anschalten des Lichtes zu beschränken (vgl. Abschnitt 5.1.1). Durch die `Monitor Agent Configuration` werden die einzelnen Komponenten gestartet, gestoppt und parametriert. Eine Referenz auf das zu aktualisierende Kontextmodell wird benötigt, um im Graphen der WB einen Einstiegsknoten für die Kontextänderungen zu haben. Der `Agent Controller` trifft außerdem die Entscheidung ein aktualisierendes Event als Symptom, beziehungsweise als die Benachrichtigung über eine signifikante Kontextänderung (vgl. Abschnitt 2.4), an den `Analyzer` weiterzugeben (siebter Schritt in Abbildung 5.5). Um die Konsistenz zwischen Realität und Modell gewährleisten zu können, ist der `Monitor` eine dauerhaft laufende Komponente, womit die PE in dieser Phase nicht zwangsläufig involviert ist.

5.1.3 Analyze

Die Analyse aufgrund eines Symptoms, befasst sich mit der Evaluation der Goals eines Prozesses, Subprozesses oder Schrittes. Nach Seiger et al. (vlg. Abschnitt 2.3, Abbildung 2.4) werden diese drei Metamodellelemente im Folgenden unter dem Begriff Process-Component (PC) zusammengefasst, um die Analyse-Phase und damit MAPE-K selbst, nicht auf die Ebene der Modellierung und -ausführung eines gesamten Prozesses zu begrenzen. Die benötigten Domain-Objekte werden in Abbildung 5.6 dargestellt, wobei der Prozess, im Gegensatz zu dem Knoten in Abbildung 5.3, zu PC verallgemeinert wurde.

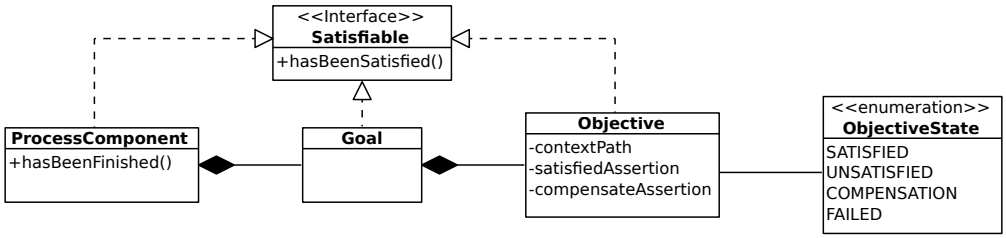

Abbildung 5.6: Domain-Modell für die Analyse-Phase

PCs, Goals und Objectives als Teil des Domain-Modells in Abbildung 5.6 sind `Satisfiable`, wodurch die Prüfung auf Erfüllbarkeit transitiv auf dem Objekt-Graphen durchgeführt werden kann. Die Unerfüllbarkeit wird durch ein `finished`-Flag in der PC signalisiert und tritt auf, wenn mindestens ein Objective eines Goals den Status `FAILED` aufweist. Sollte kein ausführbares Kommando zur Kompensation eines unerfüllten (`UNSATISFIED`) Objectives gefunden werden, ist nicht nur das Objective, sondern auch die PC unerfüllbar. Das Suchen eines solchen Befehls ist Teil der Plan-Phase und wird in Abschnitt 5.1.4 detailliert behandelt.

Zentrales Element dieses Konzepts ist die Kompensation der fehlgeschlagenen Ausführung einer PC (vgl. Abschnitt 3.6, 4.8). Eine Diskussion der Alternativen, um Zhang et al. und verteilten Transaktionsprotokollen wie XA und WS-TX, folgt in Abschnitt 5.3. MAPE-K greift reaktiv in die Prozesskonsequenzen ein, ohne den Prozess selbst zu verändern. Das Erkennen des Handlungsbedarfs geschieht über die zur PC angelegten Goals und Objectives. Die Evaluation der `compensateAssertion` signalisiert hierfür die Notwendigkeit eines Change-Requests bezüglich eines Objectives (`COMPENSATION`). `SATISFIED` ist ein Objective bei positiver Evaluation der `satisfiedAssertion`. Ein Goal ist erfüllt, wenn alle Objectives diesen Status besitzen. Der `contextPath` beinhaltet den Pfad zu einem Messpunkt innerhalb des Kontextgraphen und liefert die Variablen, gekennzeichnet durch die Raute, für die Evaluation der beiden Assertions.

Listing 5.2: Compensate- und Satisfied-Assertion mit SpEL

```
1 #objective.created.isBefore(#now.minusSeconds(2))
2 #lightIntensity > 865
```

Die erste Zeile (`compensateAssertion`), der mit SpEL[1] formulierten Assertions aus Listing 5.2, überprüft ob seit der letzten Analyse zwei Sekunden vergangen sind und ein Change-Request ausgelöst werden muss. Zeile zwei (`satisfiedAssertion`), beschreibt die Erfüllung des Objectives, wobei sich die Variable `lightIntensity` aus der Evaluation des `contextPath` ergibt. Der Ablauf einer Analyse-Phase, bezüglich der PC, innerhalb einer Iteration von MAPE-K, wird durch das Aktivitätsdiagramm in Abbildung 5.7 beschrieben. Signale, als UML-Pendant zu Events, dienen hier der Kommunikation mit anderen Komponenten und zeigen den Abbruch der Schleife und einen neuen Change-Request an. Der Datastore beinhaltet das Evaluationsergebnis zu dem Objective, bei dem die Abweichung zwischen Realität und Assertions aufgetreten ist. Ein Change-Request hält Referenzen auf das Evaluationsergebnis und das Objective. Flags auf Objekten des Domain-Modells (Abbildung 5.6) werden durch Zuweisung (`satisfied = true`) gesetzt. *Find compensable objective* iteriert über alle Objectives und filtert jene, die weder `FAILED` noch `SATISFIED` sind. *Analyze objective* evaluiert den Kontextpfad und die Assertions (vgl. Listing 5.2) um ein *evaluation result* zu erzeugen.

Da die Kontrollschleife als Teil des Kontexts betrachtet werden kann und als eigenständiges System innerhalb eines CPS agiert, werden Metriken in das Kontextmodell aufgenommen, welche die Situation der Komponente reflektieren und in die Analysephase einbezogen werden können. Daten der Systeme von Schulte et al. und Moser et al. (vgl. Abschnitt 3.1) werden in prozessexterne Objectives aufgenommen, um die infrastrukturellen Anforderungen, sowie die Selbstwahrnehmung zu unterstüt-

[1] docs.spring.io/spring/docs/current/spring-framework-reference/html/expressions.html

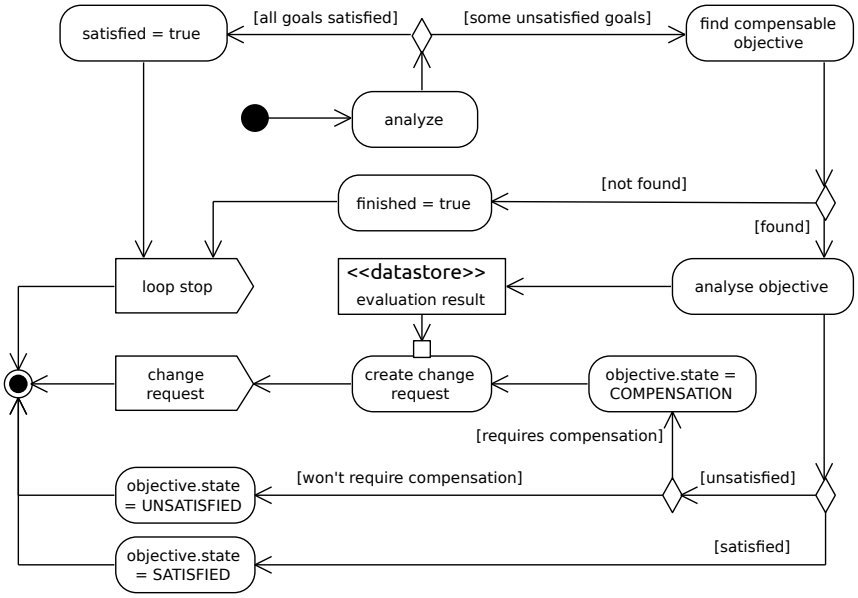

Abbildung 5.7: Analyse in einer Iteration von MAPE-K

zen (vgl. Abschnitt 4.1, 4.3). Weiterhin wird die Historie der analysierten Daten für das Lernen aus Erfahrungswissen persistiert (vgl. Abschnitt 4.7). Die durch Gurgen et al. bevorzugte, gerätespezifische Analyse (vgl. Abschnitt 4.4.4), wird in diesem Konzept nicht thematisiert, da die Autoren nicht konkret auf Funktionsweise und Schnittstellen des lokalen Reasonings und Agierens eingehen.

5.1.4 Plan

Das Planen des Handelns aufgrund eines Change-Request basiert auf der Differenz zwischen Realität und Objective, im weiteren Verlauf als *Mismatch* bezeichnet. Dieses beschreibt, im Zusammenhang mit der `satisfiedAssertion` (vgl. Abschnitt 5.1.3), das bei der Ausführung der Process-Component (PC) entstandene Problem. Change-Plans (vgl. Abschnitt 2.4) sind Kommandos, ausführbar über an die Kontrollschleife angebundene Middlewares (Abschnitt 5.1.5).

Mismatches werden durch den `Mismatch Provider`, dargestellt in Abbildung 5.8, wie folgt berechnet. Aus der `satisfiedAssertion` (vgl. Abschnitt 5.1.3), lässt sich der Zielwert auf dem Kontextpfad, sowie die logische Operation durch den *Abstract Syntax Tree (AST)* ermitteln. Die in dieser Arbeit unterschiedenen Operationen sind $>$, $<$ und $=$. So ergeben sich auch die Typen des Mismatch, *zu klein* (`TOO_LOW`), *zu groß* (`TOO_HIGH`) und *ungleich* (`UNEQUAL`). Die in Zeile zwei des Listing 5.2 dargestellte Assertion, führt bei einer `lightIntensity` von 800 Lux zu einem Mismatch vom

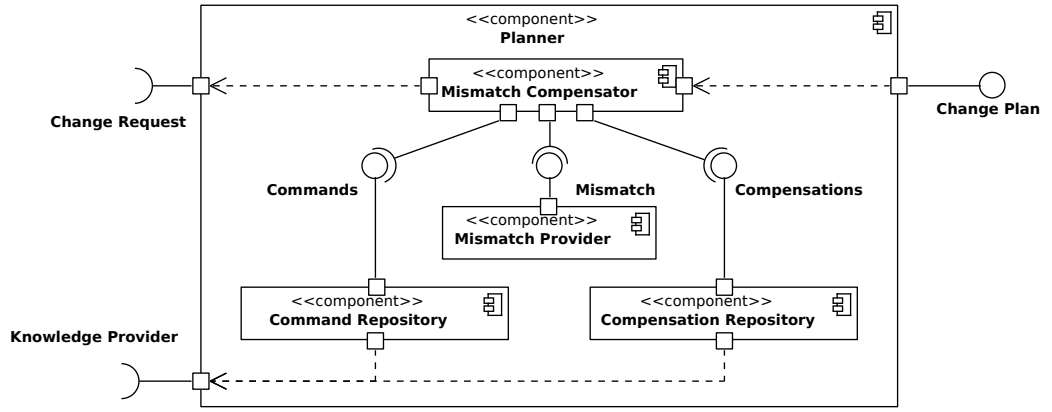

Abbildung 5.8: Detailansicht der kompensierenden Planner-Komponente

Typ `TOO_LOW`, womit *calculate mismatch* des Aktivitätsdiagramms in Abbildung 5.9 abgeschlossen ist. Ein kompensierendes Kommando wird über das `Compensation Repository` identifiziert (*find compensation candidate commands*). Kommandos besitzen die Möglichkeit einen realen Wert anzuheben (`UP`), abzusenken (`DOWN`) und zu setzen (`ASSIGN`) repräsentiert durch ihren Typen. Weiterhin werden die bereits für das Objective ausgeführten durch das `Command Repository` bereitgestellt (*find commands executed for objective*). Aus der Menge dieser, wird einer herausgefiltert, der entweder wiederholbar ist, oder noch nicht ausgeführt wurde. Ein Befehl kann wiederholt werden, wenn die Grenze des Wertebereichs nicht überschritten würde. Beispielsweise kann der Wert eines Dimmer nur bis 100 Prozent angehoben und der Lichtschalter nur einmalig ausgeschaltet werden. Die Definition von Wertebereich und dessen Überschreiten muss für die verschiedenen Zustandstypen festgelegt werden. Wenn das, nach *find command* in Abbildung 5.9, gefundene Kommando den kompensatorischen Effekt erzielen kann, wird der Zeitstempel des Objective, sowie sein Zustand neu gesetzt und ein Signal für den `Executor` ausgelöst. Andernfalls ist das Objective, das abhängige Goal und die PC fehlgeschlagen. *Verify compensating effect* greift auf den Mismatch-Typ zurück und bildet diesen, wie in Tabelle 5.1, auf das Kommando ab.

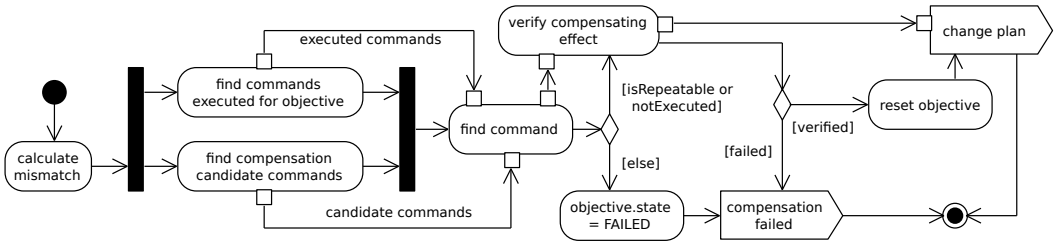

Abbildung 5.9: Planung in einer Iteration von MAPE-K

Tabelle 5.1: Mismatch-Kommando Abbildung

Typ des Mismatch	Typ des Kommandos
TOO_LOW	UP
TOO_HIGH	DOWN
UNEQUAL	ASSIGN

Wie in Abschnitt 5.1.1 erwähnt, eignet sich die DogOnt-Ontologie für die Repräsentation eines SH-Kontexts. In dieser sind neben den Räumen auch die Sensor- und Aktuator-Typen, sowie die Einheiten der Werte modelliert. Jede Instanz eines Raumes, Sensors oder Aktuators besitzt eine hierarchisch eingeordnete Typ-Relation (vgl. [CB08]). Die ID des `testNode` lokalisiert den Messpunkt innerhalb des Kontextmodells und ist damit Parameter für die Suche des `Compensation Repository` (vgl. Abbildung 5.8). Listing 5.3 repräsentiert eine Cypher-Anfrage auf dem Graphen der DogOnt-Ontologie, mittels derer Kommandos identifiziert werden, die für die Kontextmanipulation, relativ zum Messpunkt, geeignet sind. Zusammenfassend wird nach einem Aktuator gesucht, der einen dem Sensor ähnlichen Zustandstypen besitzt und sich im gleichen Raum befindet. Über diesen kann der abweichende Realwert in Anlehnung an den Mismatch manipuliert werden. Kommentare über den Zeilen des Listings beschreiben das Vorgehen detailliert. Besonderheiten befinden sich in den Zeilen zwölf und 18. Die Gleichheit der Typen des Sensors und des Aktuators (Listing 5.3, Zeile 12) ist nicht verwendbar, da die Wertebereiche der Zustände verschieden sein können. Im konkreten Fall, bezogen auf die Typhierarchie der Ontologie, besitzt ein Dimmer einen `LevelState` mit einem Wertebereich von null bis 100 Prozent. Der Lichtsensor hingegen, beschreibt einen `LightIntensityState`, der in Lux gemessen und damit nicht prozentual ausgedrückt werden kann. Da der `LightIntensityState` eine Unterklasse des `LevelState` ist (vgl. [CB08]), können die Typen mit der Toleranz einer Hierarchieebene vergleichen werden. Zeile 18 ordnet die Funktion des Aktuators der Oberklasse `ControlFunctionality` zu, ungeachtet der Typebenen.

Listing 5.3: Suche nach einem Kommando für den Mismatch

```
// Sensor (Messpunkt) besitzt einen Zustand
MATCH (sensor)-[:hasState]->(sensorState)
// Aktutator besitzt einen Zustand
MATCH (actuator)-[:hasState]->(actuatorState)
// Aktuator besitzt den Zustandswert ...
MATCH (actuatorState)-[:hasStateValue]->(actuatorStateValue)
// ... einer bestimmten Einheit (z.B. Grad-Celsius)
MATCH (actuatorStateValue)-[:unitOfMeasure]->(actuatorStateUnit)
// Sensor und Aktuator befinden sich im gleichen Raum
MATCH (sensor)-[:isIn]->(room)<-[:isIn]-(actuator)
// Typen der Zustände unterscheiden sich durch höchstens eine
    Ebene der Typhierarchie
MATCH (sensorState)-[:type]->()-[:subClassOf*0..1]->()<-[:type]-(
    actuatorState)
// Aktuator besitzt manipulierende Kommandos ...
MATCH (actuator)-[:hasFunctionality]->(function)-[:hasCommand]->(
    command)
// ... eines bestimmten Typs (z.B. 'einschalten')
MATCH (command)-[:type]->(commandType)
// Aktuator besitzt Kontrollfunktionalität (beliebig viele
    Typhierarchieebenen)
MATCH (function)-[:type]->()-[:subClassOf*]->({ name: "
    ControlFunctionality" })
// Lokalisation des Messpunktes
WHERE id(sensorState) = 666
// für Planung benötigte Rückgabewerte
RETURN DISTINCT
                        actuator.name AS actuator,
                    commandType.name AS commandType,
               command.realCommandName AS commandName,
    actuatorStateValue.realStateValue AS actuatorState,
                actuatorStateUnit.name AS actuatorStateUnit
```

Die Handlungsselektion nach Abschnitt 4.5 erfolgt hier durch Regeln, beziehungsweise den Assertions, und fallbasiertes Schließen, der Auswahl jenes Kommandos, welches einen Wert zur Kompensation von Differenzen anpasst. Konflikte in den Regeln, respektive in Goals und Objectives, müssen bei der Modellierung des Prozesses ausgeschlossen werden. Wie in Abschnitt 5.1.3 angedeutet, können Kosten und Risiken durch QoS-Attribute in die Planung, beziehungsweise die Assertions einbezogen werden. Kommandos zum Starten und Stoppen von Service-Instanzen, wie bei Schulte et al. und Moser et al., sind dafür im Kontextmodell abzubilden.

5.1.5 Execute

Mit dem Eintreffen des Change-Plan und nach Abschluss der Planungsphase, wird der kompensierende `Command` über eine Aktuator-Middleware ausgeführt. Sind verschiedene Systeme zuständig, für die Kontrolle der Aktuatoren, enthält das gewählte Kommando eine Referenz auf das anzusprechende. Die Middleware-Referenz muss im Kontextmodell festgehalten werden. Der verwendete Kanal wird durch einen `Command Executor` gekapselt, der den Befehl über das konkrete Protokoll (HTTP, WebSockets, SOAP, etc.) versendet. Das auszuführende Kommando wird in der WB mit dem jeweiligen Objective verknüpft und steht der nächsten Iteration zur Verfügung (*find commands executed for objective* in Abbildung 5.9). Nach der Ausführung, ist die `compensateAssertion` erneut zu überprüfen, wodurch die Rückkopplungsschleife geschlossen wird.

5.2 Prozessmodellierung und -ausführung

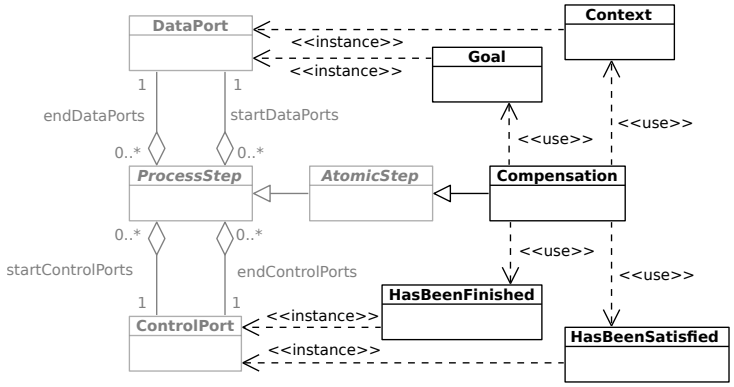

Abbildung 5.10: Erweiterung des Prozessmodells

Für die Integration von MAPE-K in das Prozessmodell nach Seiger et al., wird ein spezialisierter `AtomicStep`, wie in Abbildung 5.10, benötigt. Dieser wird parallel zu, oder unmittelbar hinter der jeweiligen Process-Component (PC) in die Modellierung eines Prozesses integriert. Die Goals und Objectives (`Goal`) um die betreffende PC werden, wie auch die Kontextreferenz (`Context`), als Ports in das Modell gegeben. Eine geeignete, standardisierte Struktur für die Repräsentation der Assertions (vgl. Abschnitt 5.1.3) und des Kontextpfads (vgl. Abschnitt 5.1.1), erlaubt die Integration des MAPE-K-Systems in bestehende Infrastrukturen und Prozessmodellierungswerkzeuge. Zwei Ausgangsports dienen dem Kontrollfluss des Prozessmodells und ermöglichen die Reaktion auf Fehlschlag (`HasBeenFinished`) und Erfolg (`HasBeenSatisfied`) der Schleife.

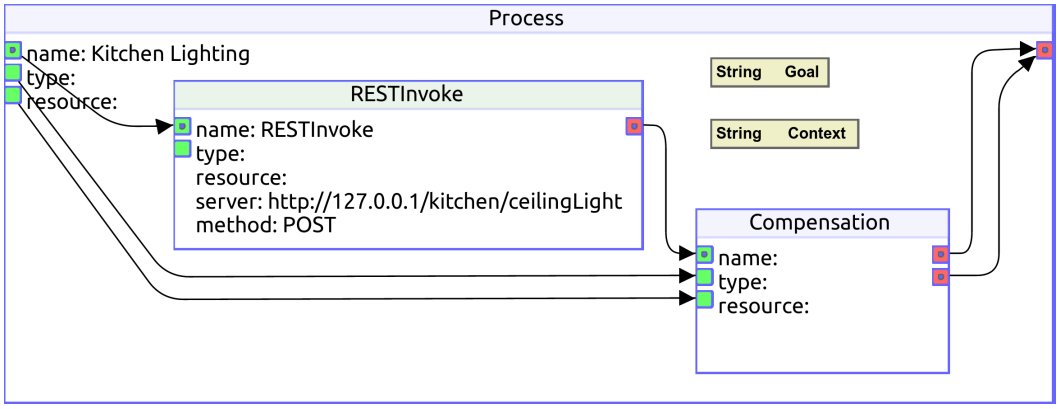

Abbildung 5.11: Beispiel für Prozessmodell mit MAPE-K-Integration

Abbildung 5.11 zeigt einen Prozess für die Regulation der Beleuchtung in der Küche eines Smart Home (SH). Modelliert wurde dieser mit einer auf dem Metamodell aus Abschnitt 2.3 basierenden Eclipse-Umgebung, bereitgestellt durch Seiger et al. Die für das Beispiel relevanten Datentypen, sind für die Ports der Erweiterung des Metamodells aus Abbildung 5.10 bestimmt. Der umschließende Prozess *Kitchen Lighting*, besitzt einen Start- und einen Endport. Sowohl die Goals, als auch die Kontextreferenz werden als Teil des Gesamtprozesses modelliert, wobei diese Daten auch von vorangegangenen PCs bereitgestellt werden können. Ziel dieses Prozesses ist das Einschalten des Lichts wodurch die Beleuchtungsintensität messbar steigt. Dafür müssen die Assertions aus Listing 5.2 und der Kontextpfad von Listing 5.1 innerhalb eines Objectives definiert werden. Das Goal zu diesem Objective wird über den dafür vorgesehenen Datenport eingebunden. Aus Gründen der Einfachheit werden die beiden möglichen Ergebnisse von MAPE-K nicht weiterverarbeitet, sondern beenden die Ausführung. `RESTInvoke` ist die Instanz einer PC des Metamodells und verantwortet HTTP-Anfragen an jene Middleware, die für das Licht in der Küche zuständig ist. Ist dieser Schritt abgeschlossen, wird der Kontrollport von `Compensation` aktiviert, die Schleife gestartet und die folgenden Aktionen ausgeführt.

1) Eine Instanz der PC wird über die `Instances`-Schnittstelle (vgl. Abbildung 5.1) in der Wissensbasis (WB) registriert.
2) Goals und Objectives werden angelegt und mit der Instanz verknüpft.
3) MAPE-K arbeitet wie in Abschnitt 5.1 beschrieben, während `Compensation` auf Rückmeldung wartet.
4) Das Event *loop stop* (vgl. Abbildung 5.7) wird emittiert und die Ergebnisflags übergeben.
5) Nach dem Löschen der Instanz der PC werden die Ausgangsports je nach Ergebnis aktiviert.

Da entweder `HasBeenFinished` oder `HasBeenSatisfied` aktiviert werden, wird der Prozess abgeschlossen. Eine Variante der Integration ist die Aufteilung dieser Schritte in eigenständige PCs. Der steigende Komplexitätsgrad im Bezug auf die Modellierung und der geringe Zugewinn an Flexibilität, sind nicht hilfreich. Jedoch kann, abhängig von der Umsetzung der Schnittstellen, dieses Konzept damit auf andere Prozessmodelle wie BPMN oder WS-BPEL übertragen werden (vgl. Abschnitt 2.2). So sind die Schritte nach dem Start der Kontrollschleife als Service-Aufrufe umsetzbar, wodurch eine Erweiterung der Modellierungssprache oder Ausführungsumgebung nicht notwendig ist.

5.3 Transaktionale Prozesseigenschaften

Transaktionalität und das ACID-Modell sind in der Domäne der Datenbanken nahezu synonym. Bei Prozessen sind die gleichen Prinzipien nicht ohne weiteres Anwendbar. Wie in Abschnitt 3.6 bereits besprochen, kann der Prozess nicht in einer einzigen Transaktion ausgeführt werden. Jeder Schritt ist als atomar zu betrachten, wodurch ein Prozess aus mehreren ACID-Transaktionen zusammengesetzt ist. Auch Konsistenz, Isolation und Dauerhaftigkeit sind nur im Bezug auf die einzelne Aktivität möglich. Schlägt nun bei paralleler Ausführung mehrerer Schritte einer fehl, müssen die anderen kompensiert werden, was nur auf einer Domain-spezifischen Ebene möglich ist und dem Rollback einer Datenbanktransaktion entspricht. Eine solche Ebene wird bei BPMN explizit, in Form kompensatorischer Aktivitäten modelliert. Die Flexibilität dieses Ansatzes steigert die Komplexität der Modellierung und ist für die grundlegende Integration von MAPE-K in CPS überflüssig. Kompensation und Wiederholung, beziehungsweise Semiatomizität nach Zhang et al., sind Bestandteil der Rückkopplung aus Abschnitt 5.1 und ermöglichen Transaktionalität beschränkt. Während BPMN imperativ beschreibt wie ein Fehler, oder eine Ausnahme auszugleichen ist, beleuchtet die Deklaration von Goals und Objectives welcher Teil des Kontexts sich auf eine bestimmte Weise verändern wird. Das Messen des Ergebnisses eines Prozessschrittes, oder das der Kompensation ist mit BPMN und vergleichbaren Prozessmodellierungs- und Ausführungssprachen nicht unmittelbar möglich (vgl. Abschnitt 2.2). Wie eine nicht zutreffende Annahme eines Prozessschritts auszugleichen ist, obliegt in diesem Konzept nicht dem Modellierenden, sondern MAPE-K.

Durch den Aspekt der Verteilung von CPS-Komponenten, liegt die Verwendung von XA bei der Ausführung von Prozessen nahe (vgl. Abschnitt 3.6). Wie auch bei den ACID-Prinzipien, birgt die Laufzeit einer Prozesstransaktion und das damit verbundene Sperren von Ressourcen das Problem. Eine Variante für die Integrati-

on von XA ist die Verwendung von *soft-Locks*, bei denen die Sperre nicht beliebig lang aufrecht erhalten werden kann. Die zwangsweise Unterstützung von XA durch alle Subsysteme, erschwert das Einbinden dieses Protokolls zusätzlich, kann aber durch den Einsatz geeigneter Middlewares, über dieses Konzept hinaus, durchgesetzt werden. WS-TX unterliegt den gleichen Prinzipien wie XA und unterstützt Service-orientierte Architekturen, womit dessen Einsatz für Prozesse in CPS noch zu untersuchen ist.

Der Einsatz der in diesem Konzept integrierten Kontrollschleife ermöglicht Prozesstransaktionalität durch Semiatomizität nach Zhang et al. ohne die Verwendung zentraler Schritte (vgl. Abschnitt 3.6).

5.4 Simulation und Vorhersage

Die Rückkopplungsschleife kann auf der Ebene des Prozesses, eines Subprozesses, oder der des einzelnen Schrittes agieren. Rückers System benötigt einen vollständigen Prozess (vgl. Abschnitt 3.7), wodurch der Einsatz von MAPE-K in diesem Zusammenhang eingeschränkt ist. Da die Iterationen der Schleife zeitlich mit der Realität verknüpft, die Simulationen aber nicht auf Echtzeit ausgelegt sind, ist das Ergebnis nicht zuverlässig zum Zeitpunkt des Planens verfügbar. Ein Simulationsresultat lässt sich nur zeitunabhängig in eine unbestimmte folgende Iteration integrieren, was den Anforderungen aus Abschnitt 4.4.1 und Abschnitt 4.4.4 widerspricht. Die Abbildung der fortschreitenden Realzeit auf Iterationen von MAPE-K ermöglicht Simulation mit Rückkopplung, kollidiert aber ebenfalls mit den genannten Anforderungen. DISMO, als ein generischer Ansatz zur Simulation (vgl. Abschnitt 3.7), ist auf die Vollständigkeit des Prozesses nicht angewiesen und somit auf jede der drei Ebenen anwendbar. Allerdings bestehen die Schwierigkeiten mit Echtzeitanalyse und -planung auch bei diesem System.

CPS-Prozesse beeinflussen maßgeblich einen realen Kontext. Eine Simulation auf diesem Kontext wäre demnach ein Experiment, ohne Isolation des Prozesses, dessen Variablen, Aktuatoren und Sensoren. Sowohl Simulation, als auch daraus zu ziehende Schlüsse und Vorhersagen, sind in Verbindung mit MAPE-K in diesem Konzept, während der Prozessausführung, nicht möglich.

6 Implementation

Nach der Vorstellung des Konzepts im vorangegangenen Kapitel, beschreibt dieses die Umsetzung eines vertikalen Prototyps. Neben den verwendeten Technologien, werden die Architektur des Systems, Schnittstellen zwischen den Komponenten, die MAPE-K-Schleife und deren Integration in die bestehende Ausführungsumgebung von Seiger et al. besprochen.

6.1 Verwendete Technologien

Die Umsetzung der Komponenten der Kontrollschleife, wie auch die von PROtEUS, wurde ausschließlich mit Java 8 bewerkstelligt. Verschiedene auf XML und der *JavaScript Object Notation (JSON)*[1] basierende Austauschformate ermöglichen die Interoperabilität der Module.

PROtEUS ist die von Seiger et al. entwickelte Ausführungsumgebung für Prozesse in CPS [SHS15]. Die Prozessengine und das in Abschnitt 2.3 besprochene Metamodell basieren auf dem EMF und dessen Ecore-Modellen, welche auf die MOF aufbauen. Für die graphische Modellierung eines Prozesses für PROtEUS, steht das Eclipse-Tool Graphiti[2] zur Verfügung. Aufrufe an externe Services werden durch die *Open Services Gateway initiative (OSGi)*[3] und von Seiger et al. implementierte Adapter für SOAP, *Representational State Transfer (REST)* und *XML Remote Procedure Call (XML-RPC)* ermöglicht [SHS15].

Spring ist ein in Java geschriebenes Open-Source Framework für die Entwicklung und Konfiguration von Enterprise-Applikationen aller Art. Vergleichbar mit dem

[1] www.json.org
[2] www.eclipse.org/graphiti
[3] www.osgi.org

Ansatz der *Enterprise JavaBeans (EJB)*[1], bietet es die Grundlagen für Dependency Injection[2], Transaktionsmanagement, Web-Applikationen, Datenzugriffsabstraktion, Messaging und ausführliches Testing[3]. Für das Verwalten von Abhängigkeiten und Ausliefern von Softwarekomponenten oder Services, ist Spring Boot[4] geeignet. Ein eingebetteter Web-Server erlaubt unabhängige Artefakte. Die Konfiguration wird zu einem großen Teil von den Komponenten dieses Projektes übernommen und vereinfacht die Entwicklung der Anwendungslogik. Sie kann außerdem durch die *Spring Expression Language (SpEL)*[5] angereichert werden. Traversieren und Manipulieren des Objekt-Graphen einer Spring-Anwendung sind mit dieser Sprache zur Laufzeit möglich. Sie eignet sich weiterhin zur Evaluation boolescher Ausdrücke. Spring Data REST[6] legt eine Zugriffsschicht von Hypermedia-getriebenen HTTP-Ressourcen um das Domain-Modell und bietet eine Realisierung des Anwendungs-API. Das Spring-Projekt hängt nicht ausschließlich von seiner großen und ständig wachsenden Community ab, sondern wird außerdem von der Firma Pivotal[7] vorangetrieben.

Neo4j ist eine der führenden Graphdatenbanken[8]. Die in Java implementierte Open-Source-Lösung wird von Neo Technology entwickelt. Das Abfragen und die Manipulation der Daten geschieht bevorzugt über die proprietäre Sprache *Cypher*[9], wobei Alternativen verfügbar sind[10]. Auf der GraphConnect 2015 in San Francisco wurde außerdem *openCypher*[11] vorgestellt, womit eine quelloffene Standardisierung von Cypher begonnen hat. Durch die starke Verbreitung und Popularität existiert eine aktiv entwickelte Anbindung an das Spring Framework im Rahmen des Spring Data Projekts[12].

OpenHAB bündelt unterschiedliche Heimautomationssysteme und -Technologien in einer Middleware, die umfassende Automatisierungsregeln und eine einheitliche Nutzerschnittstelle ermöglicht[13]. Sensordaten werden aggregiert und über ein

[1] www.java.net/projects/ejb-spec
[2] Zuweisung von Abhängigkeiten eines Objekts durch externe Instanz
[3] www.spring.io
[4] projects.spring.io/spring-boot
[5] docs.spring.io/spring/docs/current/spring-framework-reference/html/expressions.html
[6] projects.spring.io/spring-data-rest
[7] www.pivotal.io
[8] www.neo4j.com
[9] neo4j.com/docs/stable/cypher-introduction.html
[10] Gremlin Graph Query Language (gremlin.tinkerpop.com)
[11] www.opencypher.org
[12] projects.spring.io/spring-data-neo4j
[13] www.openhab.org/features/introduction.html

REST-API bereitgestellt. Über dieses ist es auch möglich Aktuatoren anzusprechen, oder Befehle auszulösen.

Eine Erweiterung von OpenHAB entstand mit der Bachelorarbeit von André Kühnert [Küh15]. Mittels der DogOnt-Ontologie (vgl. Abschnitt 5.1.1) erarbeitete er eine semantische Schicht (Semantic Access Layer), die das formalisierte Einbetten von Sensoren und Aktuatoren in den Kontext eines Smart Home erlaubt. Der daraus hervorgegangene Prototyp wurde in dieser Arbeit als Quelle des Kontextmodells verwendet. Zur Umwandlung der Ontologie in einen Graphen wurde Sesame[1], ein Java-Framework zur Verarbeitung von RDF-Daten, verwendet. Es besitzt Adapter für alle gängigen Austauschformate des RDF, einschließlich der *Web Ontology Language (OWL)*[2] und Turtle[3].

Weiterhin verwendet wurden *Elasticsearch*[4] für die Indexierung historischer Kontextdaten und *Jackson*[5] zur Serialisierung und Deserialisierung der Domain-Objekte.

6.2 Architektur und Schnittstellen

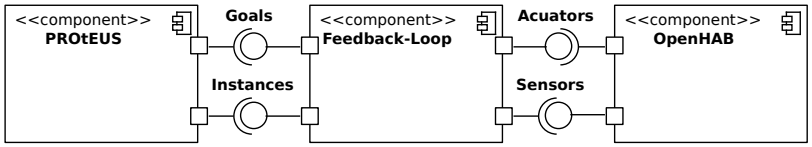

Abbildung 6.1: Übersicht der prototypischen Komponentenstruktur

In Übereinkunft mit den Anforderungen aus Abschnitt 4.1 und Abschnitt 4.2, ist die Komponente `Feedback-Loop` ein Web-Service mit einem REST-API, das JSON als primäres Austauschformat benutzt. Service-Aufrufe innerhalb eines Prozesses erlauben so eine nicht-invasive Methode der Integration von MAPE-K. Wie in Abbildung 6.1 zu sehen, benötigt der `Feedback-Loop` die Goals und Prozessinstanzinformationen von PROtEUS. In den folgenden Listings, wird eine MAPE-K-Service-Instanz mit der Adresse *http://smart.home* vorausgesetzt.

Instances. Da der Prozess lediglich eine mögliche Anbindung der MAPE-K-Komponente ist (vgl. Abschnitt 5.1.3), wird im weiteren Verlauf von Process-Component

[1] www.rdf4j.org
[2] www.w3.org/TR/owl-ref
[3] www.w3.org/TR/turtle
[4] www.elastic.co/products/elasticsearch
[5] wiki.fasterxml.com/JacksonHome

(PC) gesprochen. Wird eine PC gestartet, muss der MAPE-K-Komponente der Name und ein Kontext, innerhalb dessen diese ausgeführt wird, mitgeteilt werden. Die HTTP-Anfrage ist in Listing 6.1 dargestellt. Die angelegte PC entspricht dem `P` aus Abbildung 5.3 in der Wissensbasis (WB). Die Relation `runsWithin` wird automatisch anhand der Kontextreferenz *http://smart.home/contexts/3132* durch Spring Data REST angelegt.

Listing 6.1: HTTP-Anfrage zum Erzeugen einer Process-Component Instance

```
1 POST /workflows HTTP/1.1
2 Host: smart.home
3 Content-Type: application/json
4
5 {
6   "name": "prepare cooking with friends",
7   "context": "http://smart.home/contexts/3132"
8 }
```

Goals. Eine Process-Component (PC) (vgl. Abschnitt 2.3) repräsentiert einen Prozess, einen Subprozess oder einen einzelnen Schritt, der mit einem Goal versehen werden kann. Die in Listing 6.2 dargestellte HTTP-Anfrage beinhaltet neben dem Namen, die Referenz zu der betreffenden PC. Das in diesem Fall einzige Objective beschreibt eine `satisfiedExpression` (Zeile zehn), nach der die Variable `lighting` mit einem Wert größer als 865 belegt sein muss, damit das Objective erfüllt ist. In der darauf folgenden Zeile wird die `compensateExpression` definiert, anhand derer die Notwendigkeit der Kompensation erkannt wird. Bei diesem Beispiel müssen dafür zwei Sekunden seit Erzeugen des Objectives vergangen sein. Diese beiden Expressions sind Synonyme für die in Abschnitt 5.1.3 besprochenen Assertions. Das in Zeile 13 befindliche, rein technisch motivierte Feld `testNodeIdExpression` dient der Identifikation des Messpunktes innerhalb des Kontextpfads, da die `contextExpression` mehrere Variablen zurückgeben kann. Aufgrund mehrerer möglicher Sensoren dieses Pfads, wird der Durchschnittswert (`avg`) der Werte berechnet (Zeile 19) und die erste `stateId` als Messpunkt zurückgegeben (Zeile 20).

Listing 6.2: HTTP-Anfrage zum Erzeugen eines Goals

```
1 POST /goals HTTP/1.1
2 Host: smart.home
3 Content-Type: application/json
4
5 {
6   "name": "enough light for cooking",
7   "workflow": "http://smart.home/workflows/3133",
8   "objectives": [{
9     "name": "kitchen light > 865 lux within two seconds",
```

```
10      "satisfiedExpression": "#lighting > 865",
11      "compensateExpression":
12          "#objective.created.isBefore(#now.minusSeconds(2))",
13      "testNodeIdExpression": "#stateId",
14      "contextExpressions": [
15        "MATCH (t)-[:type]->(s { name: 'LightSensor' })",
16        "MATCH (t)-[:isIn]->(r { name: 'Küche' })",
17        "MATCH (t)-[:hasState]->(s:LightIntensityState)",
18        "MATCH (s)-[:hasStateValue]->(v)",
19        "RETURN avg(toFloat(v.realStateValue)) AS lighting, ",
20                        "head(collect(id(s))) AS stateId"
21      ]
22    }]
23  }
```

Actuators und Sensors. Das OpenHAB-API ist eine JSON- und XML-basierte REST-Schnittstelle[1]. Diese wurde im Prototypen durch den `OpenHabMonitorAgent`, beziehungsweise den `OpenHabCommandExecutor` gekapselt. Ersterer fragt mit einer definierten Frequenz die Items, die Zustandswerte der Sensoren und Aktuatoren ab (*Polling*), wandelt die JSON-Repräsentation in ein internes Format (`OpenHabItem`), filtert und überprüft die Veränderungen. Details, unter anderem um die Funktionsweise des Filters, dieses Monitor-Agents werden in Abschnitt 6.3 besprochen. Zweiterer ist Zuständig für das Ausführen von Befehlen (`Command`) aufgrund eines Change-Plan und gibt ein HTTP-Post auf den Endpunkt des betreffenden Aktuators ab. Die Kommunikationsbasis zwischen `Feedback-Loop` und `OpenHAB` wird durch Feign[2], einen deklarativen HTTP-Client von Netflix[3], bereitgestellt.

Ein weiterer, nicht abgebildeter Kommunikationskanal übermittelt der PC die Fertigstellung der Kontrollschleife mittels Server-Sent Events (SSE). Wie in Abschnitt 5.2 besprochen, zeigen boolesche Werte den positive (`HasBeenSatisfied`) oder negativen (`HasBeenFinished`) Ausgang der Kompensation. Im Prozessmodell kann über Kontrollports entsprechend reagiert werden.

[1] github.com/openhab/openhab/wiki/REST-API
[2] github.com/Netflix/feign
[3] netflix.github.io

6.3 Die Komponente der Rückkopplung

Davon ausgehend, dass der Kontext definiert und eine PC mit Goals und Objectives angelegt wurde, ist dieser Abschnitt den Implementierungsdetails und Zuständigkeiten der Komponente des `Feedback-Loop` (vgl. Abbildung 6.1, 5.1) gewidmet. Abbildung 6.2 vermittelt hierbei einen Überblick, der im weiteren Verlauf zur Erklärung referenziert wird. Das gesamte Projekt ist modular angelegt und in den Service, dessen API und Plugins, getrennt nach Zuständigkeiten, aufgeteilt. Zur Laufzeit werden die API-Implementationen von einem `ProteusFeedbackPlugin` ausgewählt und parametriert. Der Service hat zum einen die Aufgabe die externen Schnittstellen bereitzustellen, zum anderen Verwaltet er die laufenden Kontrollschleifen. Jede Phase von MAPE-K wird durch ein Interface und dessen korrespondierende Implementation abgebildet (*Loop*-Schicht in Abbildung 6.2).

6.3.1 Knowledge

Operationen auf den Entitäten, wie das Speichern, Lesen, Aktualisieren oder Löschen (Create, Read, Update, Delete (CRUD)) von Kontextmodellen, PCs und Goals, werden von Spring Data REST und Spring Data Neo4j ohne manuelles Zutun abgedeckt. So müssen für das Bereitstellen der `Knowledge` lediglich Interfaces definiert und annotiert werden, durch die das Framework Persistenz (`GraphRepository`) und ein REST-API (`RepositoryRestResource`) bereitstellen kann. Listing 6.3 zeigt das Repository für die Entität `Context`. Durch die Erweiterung des `GraphRepository`, beinhaltet das `ContextRepository` die für die Verwaltung der Kontextmodelle notwendigen Operationen. Für komplexere Anfragen an die Wissensbasis existiert mit der *query*-Methode von `Neo4jOperations` eine native Schnittstelle.

Listing 6.3: Kontext-Repository mit Spring Data Neo4j+REST

```
1 @RepositoryRestResource
2 public interface ContextRepository
3         extends GraphRepository<Context> {}
```

6.3.2 Monitor

Das Monitoring von Kontexten startet sowohl nach der Initialisierung des Service, als auch nach der Fertigstellung des Imports, beziehungsweise dem Übertragen der DogOnt-Ontologie in das Neo4j-basierte Modell. Jedes dieser Kontextmodelle erhält eine eigene Instanz des `RealTimeAgentMonitor`. Diese Implementie-

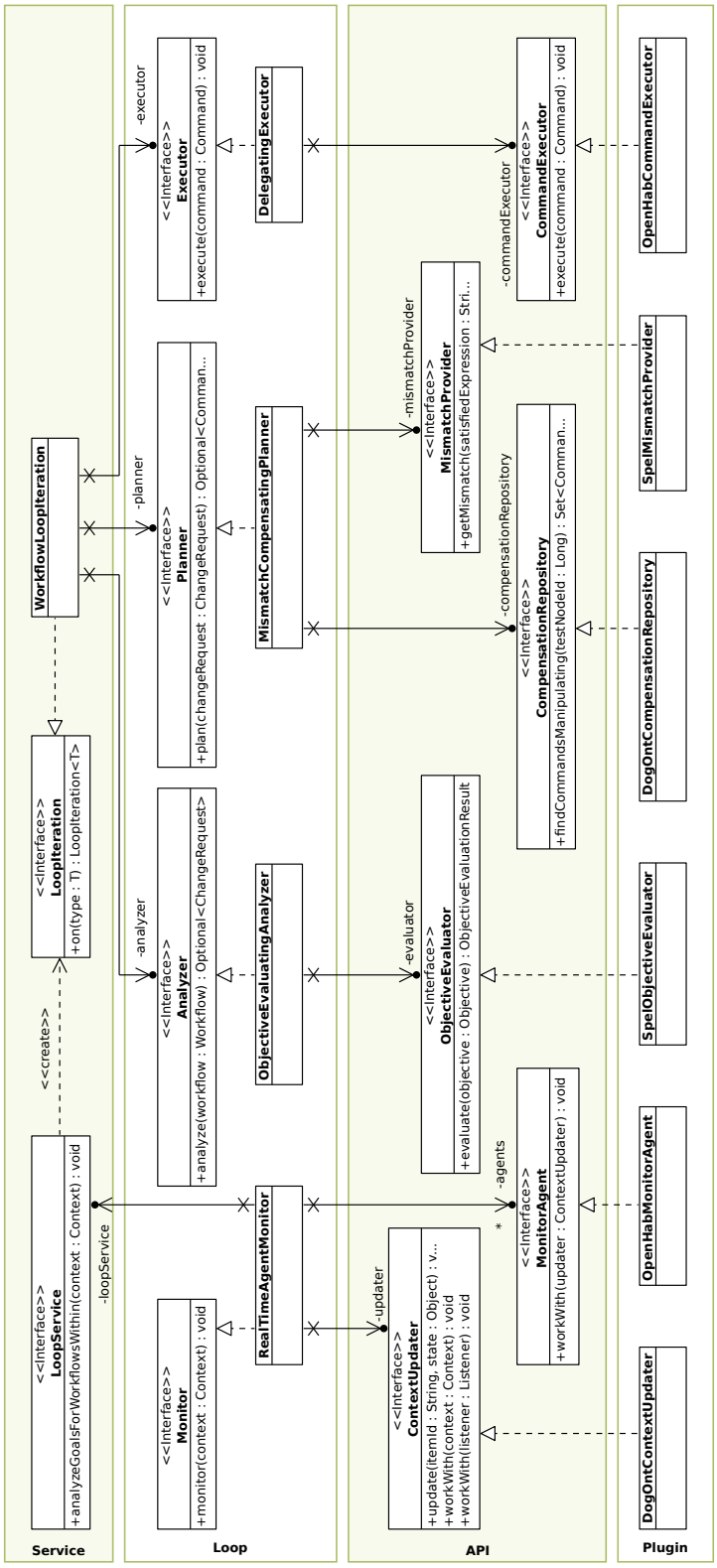

Abbildung 6.2: Implementierungsdetails des Feedback-Loops

rung des `Monitor`-Interface, basiert auf der Integration verschiedener Agents (vgl. Abschnitt 5.1.1) und der Abbildung der realen Zeit auf das Modell. Der in diesem Prototyp implementierte `OpenHabMonitorAgent` stellt die Verbindung zu einer OpenHAB-Middleware her und informiert bei einer signifikanten, für ein Symptom verantwortlichen, Kontextveränderung den `ContextUpdater`, welcher durch den `DogOntContextUpdater` im DogOnt-Plugin implementiert ist. Diese Indirektion ist aufgrund des DogOnt-spezifischen Kontexts notwendig (vgl. Abschnitt 5.1.2). Weiterhin wird dieser `ContextUpdater` jeder Sekunde mit der Veränderung der Zeit konfrontiert.

Die Signifikanz einer Veränderung des Kontexts wird über einen parametrierbaren Schwellwert bestimmt, der für alle Sensoren gleichermaßen gelten soll. Da die Wertebereiche der Sensoren sich stark unterscheiden, muss ein empfangener Wert für den Vergleich mit diesem Schwellwert normalisiert werden. Wenn ein Wert im `OpenHabMonitorAgent` eintrifft wird überprüft, ob er größer als das bis dahin erreichte Maximum ist und gegebenenfalls als das neue übernommen. Die Division des erfassten Datums durch diesen zeitlich relativ größten Sensorwert, entspricht der Normalisierung des Wertebereichs. Signifikanz ist erreicht, wenn der Betrag der Differenz des aktuellen und des letzten Wertes größer als der Schwellwert ist. `SignificanceFilter` implementiert diesen Algorithmus, nachzulesen in Listing 6.4. Zeile 14 bezieht sich dabei auf den Spezialfall der Gleichheit von Wert und Maximum.

Listing 6.4: Signifikanztest für Sensorwerte

```
1  boolean isSignificant(String state, String name) {
2      double number = Double.valueOf(state);
3      double last = normal.getOrDefault(name, Double.MIN_VALUE);
4      double max = maximum.getOrDefault(name, Double.MIN_VALUE);
5
6      if (max < number) {
7          maximum.put(name, number);
8          max = number;
9      }
10
11     double current = (number / max);
12     normal.put(name, current);
13
14     if (current == last && last == 1.0) return true;
15     else return abs(current - last) > delta;
16 }
```

6.3.3 Analyze

Sobald ein Symptom auf einem Kontextmodell entdeckt wurde, ist der `LoopService` verantwortlich für das Erzeugen einer MAPE-K-Iteration (`WorkflowLoopIteration`). Im Zuge dessen, werden alle Goals aller Process-Components (PCs), die innerhalb dieses Kontexts ablaufen (`runsWithin`), durch den `Analyzer` auf ihre Erfüllung überprüft, wodurch ein Change-Request ausgelöst werden kann. Die Implementierung `ObjectiveEvaluatingAnalyzer`, dargestellt in Listing 6.5, prüft im ersten Schritt, ob die Ziele einer PC erfüllt sind (Zeile 1-2). Weiterhin kann die Kompensation durch das Fehlen geeigneter Objectives beendet werden (vgl. Abschnitt 5.1.3). Dieser Fall tritt ein, wenn weder unerfüllte noch in der Kompensation befindliche Objectives verfügbar sind (Zeile 4-9). Wenn alle Goals erfüllt sind, oder die Kompensation nicht fortgeführt werden kann, wird die PC als beendet markiert und ein leerer Change-Request zurückgegeben (Zeilen 11-13). Andernfalls sucht der Algorithmus anhand der Objectives nach einem geeigneten (Zeile 15-19).

Listing 6.5: Analyse der Process-Component-Goals

```
1  boolean allGoalsSatisfied = workflow.getGoals().stream()
2      .allMatch(Goal::hasBeenSatisfied);
3
4  boolean nothingLeft = !workflow.getGoals().stream()
5      .filter(goal -> goal.getObjectives().stream()
6          .anyMatch(o -> (o.getState() == COMPENSATION) ||
7                         (o.getState() == UNSATISFIED)))
8      .findAny()
9      .isPresent();
10
11 if (allGoalsSatisfied || nothingLeft) {
12     workflow.setFinished(true);
13     return Optional.empty();
14 } else {
15     return workflow.getGoals().stream()
16         .flatMap(goal -> goal.getObjectives().stream())
17         .map(this::toChangeRequest)
18         .filter(Objects::nonNull)
19         .findAny();
20 }
```

In Listing 6.6 wird der `ObjectiveEvaluator` für das Erkennen der Erfüllung eines Objectives und der Notwendigkeit einer Kompensation genutzt. Die auf der SpEL basierende Implementierung (`SpelObjectiveEvaluator`), führt den Cypher-Query (`contextExpression`) auf dem Kontextmodell aus (Zeile 1) und erhält damit die Variablen zur Evaluation der Assertions `satisfiedExpression` und `compensateExpression` (Zeile 2-3, vgl. Listing 6.2). Sollte keine Erfüllung eingetreten, die Notwendigkeit

der Kompensation jedoch gegeben sein, wird ein Change-Request erstellt und das Objective in den korrespondierenden Zustand versetzt. In den anderen beiden Fällen (Zeile 8-13), ist die Kompensation nicht relevant und es erfolgt lediglich eine Aktualisierung des Zustands.

Listing 6.6: Umwandlung von Objective zu Change-Request

```
1  ObjectiveEvaluationResult result = evaluate(objective);
2  boolean weCantGetNoSatisfaction = !result.hasBeenSatisfied();
3  boolean compensationIsRequired = result.shouldBeCompensated();
4
5  if (weCantGetNoSatisfaction && compensationIsRequired) {
6      objective.setState(State.COMPENSATION);
7      return ChangeRequest.on(objective, result);
8  } else if (weCantGetNoSatisfaction) {
9      objective.setState(State.UNSATISFIED);
10     return null;
11 } else {
12     objective.setState(State.SATISFIED);
13     return null;
14 }
```

6.3.4 Plan

Ein generierter Change-Request wird an die `Planner`-Implementation übergeben und mittels eines zum Mismatch passenden Kommandos für die Kompensation vorbereitet. Das `commandRepository` ist dabei die Instanz eines `GraphRepository<Command>` für CRUD-Operationen auf Kommandos (vgl. Abschnitt 6.3.1). Listing 6.7 zeigt die unmittelbare Implementation des Konzepts der Planungsphase (vgl. Abschnitt 5.1.4, Abbildung 5.9). Die Methode `isCompensating` der Zeile zwölf überprüft die durch Tabelle 5.1 gegebene Abbildung. `prepared` (Zeile 16) verknüpft das als Kompensation identifizierte Kommando mit dem Objective, markiert es als unerfüllt und erneuert den Zeitstempel für die nächste Iteration.

Listing 6.7: Planen der Kompensation

```
1  Set<Command> executed = commandRepository
2      .findCommandsExecutedFor(objective);
3
4  Set<Command> compensations = compensationRepository
5      .findCommandsManipulating(testNodeId);
6
7  ContextMismatch mismatch = mismatchProvider
8      .getMismatch(satisfiedExpression, contextVariables);
9
```

```
10 Optional<Command> command = compensations.stream()
11     .filter(it -> it.isRepeatable() || !executed.contains(it))
12     .filter(it -> isCompensating(mismatch, it))
13     .findAny();
14
15 if (command.isPresent())
16     return Optional.of(prepared(command.get(), objective));
17 else
18     objective.setState(Objective.State.FAILED);
```

6.3.5 Execute

Die Ausführung des Kommandos zur Kompensation eines fehlgeschlagenen Objectives, obliegt einem `CommandExecutor`. Implementiert wird dieser im Prototyp durch den durch das OpenHAB-Plugin bereitgestellten `OpenHabCommandExecutor`. Das REST-API der verwendeten Middleware referenziert Aktuatoren und Sensoren über den URI. Ein HTTP-Post mit dem Namen des Kommandos auf diesen, ermöglicht die Manipulation des Aktuatorenzustands. Alle Befehlsnamen sind durch die DogOnt-Ontologie im Kontextmodell hinterlegt (vgl. `realCommandName` des Listing 5.3 aus Abschnitt 5.1.4). Sollte während der Ausführung ein Fehler auftreten und eine Exception geworfen werden, wird das Kommando als nicht wiederholbar markiert und kann in folgenden Iterationen nicht als Kompensationsbefehl genutzt werden (vgl. Abschnitt 6.3.4).

6.4 Integration in die Ausführungsumgebung

Als Web-Service implementiert, ließe sich der Prototyp ohne Veränderung der Process-Engine (PE) integrieren (vgl. Abschnitt 6.2). Da der resultierende Prozess in diesem Fall die Business-Logik an Komplexität übertreffen kann, wurde auf Basis des `LoadClassStep` ein eigener Prozessschritt umgesetzt. Diese Grundlage erlaubt beliebigen Java-Code als Teil eines Prozesses auszuführen und bietet somit eine generische Methode der Erweiterung des Prozessmodells von PROtEUS. Eingehende Ports dienen der Konfiguration der Kompensation und beinhalten, neben `Context` und `Goal` (vgl. Abschnitt 5.2), den URI des MAPE-K-Service und den Namen der zu überwachenden Process-Component (PC). Wie im Konzept besprochen, sind die ausgehenden Ports `HasBeenFinished` und `HasBeenSatisfied`.

Wird der `LoadClassStep` in einer Prozessmodellinstanz ausgeführt, wird die Verbindung zum MAPE-K-Service hergestellt. Das betrifft sowohl den REST-HTTP-Client, als auch einen für die SSE. Über den in Abschnitt 6.2 vorgestellten HTTP-Request (Listing 6.1), wird die Instanz der PC registriert. Danach werden die Ports vom Typ `Goal` gesammelt, deren Daten an den Service übertragen (Listing 6.2) und damit die PC-Instanz verknüpft. Die Kontrollschleife arbeitet eigenständig und sendet im Falle des Abschlusses die Repräsentation der PC über den SSE-Kanal an den `LoadClassStep`. Entsprechend der Auswertung dieser Werte werden die Ports gesetzt und der Prozessschritt nach dem Löschen der Prozessbezogenen Daten beendet.

7 Evaluation

Die Evaluation des Konzepts, wie auch des Prototypen, wird durch dieses Kapitel beschrieben. Ein künstliches Szenario bietet die Grundlage für das Überprüfen der zu erwartenden Funktion der Implementierung. Im weiteren Verlauf werden die Anforderungen geprüft und aufgetretene Probleme diskutiert. Abschließende Betrachtungen ordnen diese Arbeit in ihren wissenschaftlichen Kontext ein und klären offene Fragen.

7.1 Laborexperiment

Ein Server beinhaltet in Ausführungsumgebung von PROtEUS, sowie OpenHAB- und den MAPE-K-Service. Alle in diesem Szenario verwendeten Nachrichten an den Web-Service sind in Abschnitt A.1.3 nachzulesen. Wie im Konzept angedeutet, ist das Ziel des beispielhaften Prozesses in Abbildung 7.1 die Regulation der Beleuchtung (vgl. Listing 5.1 in Abschnitt 5.1.1).

Der in Listing A.1.3 dargestellte HTTP-Request legt ein Goal an, durch welches innerhalb von zwei Sekunden eine Beleuchtungsintensität von über 865 Lux in der Küche gemessen werden muss. Diese *Küche* war ein Laboraufbau mit zwei gedimmten Lampen und zwei Lichtsensoren, die in dem Messpunkt des Kontextpfads über ihren Mittelwert vereinigt wurden (Zeile 19 von Listing A.1.3). Ein in Abbildung 7.1 gezeigter `RESTInvoke`-Schritt schaltet die erste Lampe ein. Insgesamt war das System so abgestimmt, dass eine Lampe nicht genügen würde, um die 865 Lux zu erzeugen. Somit musste die Rückkopplungsschleife eingreifen, die zweite Lampe innerhalb des Kontextmodells identifizieren und den korrekten Befehl an OpenHAB übermitteln. Während der Ausführung des Prozesses erkannte der MAPE-K-Service den Mismatch, fand den Dimmer und das Kommando zum Inkrementieren der Position. In Schritten von fünf Prozent (OpenHAB spezifisch), mit einem Abstand von zwei Sekunden (vgl. `compensateExpression`, Zeile 11-12 in Listing A.1.3), wurde das Licht der zweiten Lampe verstärkt. Die Kompensationsdauer war erwartungsgemäß und das Experiment erfolgreich.

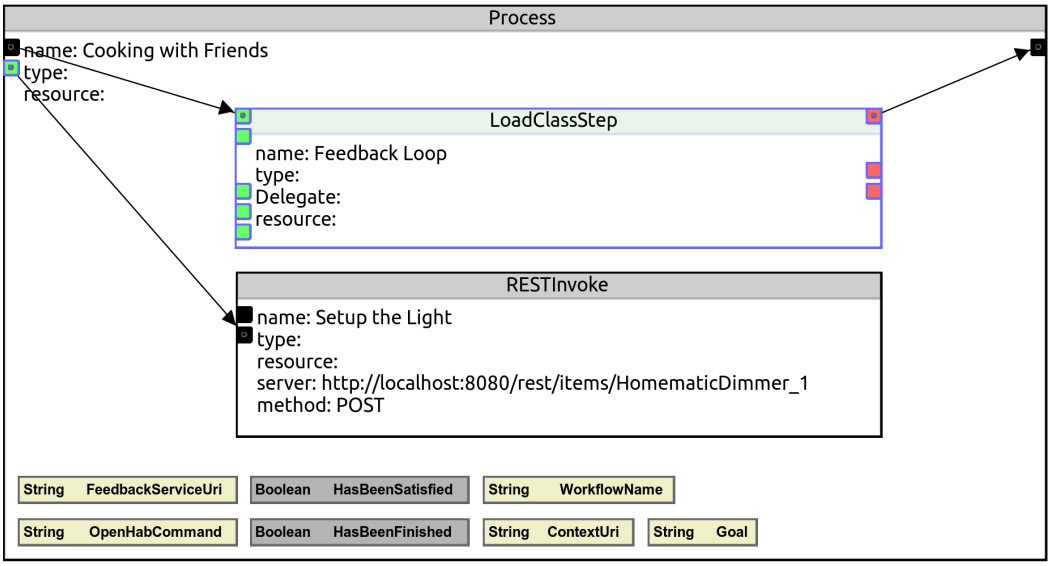

Abbildung 7.1: Für Laborexperiment verwendeter Prozess

7.2 Anforderungsabdeckung

In folgender Tabelle 7.1 werden die Anforderungen und deren Erfüllung in Konzept (K) und Prototyp (P) zusammengefasst. Wie in Abschnitt 4, ist das tertiäre Beurteilungssystem in *erfüllt*, *unerfüllt* und *beschränkt* ((+)) aufgeteilt, wobei eine Beschränkung hier in der noch ausstehenden Anbindung an ein Fremdsystem, oder einer Erweiterung besteht.

Tabelle 7.1: Anforderungsabdeckung bzgl. Konzept und Prototyp

Anforderung	K	P	Erklärung
Infrastruktur			
modular	+	+	Pluginbasierter Web-Service (Microservice[a])
fehlertolerant	(+)	-	benötigte höhere Abstraktionsschicht (z.B. PaaS[2]); tolerant bzgl. Mismatch und dessen Kompensation

[1] de.wikipedia.org/wiki/Microservices
[2] Platform as a Service (vgl. de.wikipedia.org/wiki/Platform_as_a_Service)

Anforderung	K	P	Erklärung
elastisch	(+)	-	möglich mittels Replikation und Clustering
Interoperabilität			
stand. Schnittstellen	+	+	Web-Service mit JSON-API
Selbstwahrnehmung			
Situation	(+)	-	möglich durch prozessexterne Objectives oder einen dedizierten Wahrnehmungsprozess
Zustand	(+)	-	möglich durch prozessexterne Objectives oder einen dedizierten Wahrnehmungsprozess
Handlungsoptionen	+	+	durch Inferenz/Traversieren auf Kontextmodell
Aggregation			
parallel	+	+	parallel arbeitende Monitor-Agents
echtzeitlich	+	+	Realzeit durch Agent-Monitor (`RealTimeAgentMonitor`), abfragen der Werte min. im Sekundentakt
von Rohdaten	+	+	mittels Kapselung durch Agents (hier `OpenHabMonitorAgent`)
fusionierter Daten	+	+	Abstraktion des Agent-Monitor homogenisiert Event-Quellen
Modell			
generisch	+	+	schemaloses Graphmodell, Struktur durch DogOnt
semantisch	+	+	Graphtraversion und Inferenz

Anforderung	K	P	Erklärung
erweiterbar	+	+	beliebig erweiterbare Kontextstruktur
Konsistenz			
transaktionale Kontextmanipulation	(+)	(+)	Prozess wird durch Kontrollschleife transaktionalisiert (ohne *Rollback*)
variabler Transaktionalitätsgrad	+	+	abhängig von Goal-Definition bei der Modellierung
Analyse			
echtzeitlich	+	+	mit Auftreten eines Symptoms (Echtzeitmonitoring)
bevorzugt gerätespezifisch	-	-	fehlende Standards und Technologien
Evaluation Situation	(+)	-	möglich durch Daten der Systeme von [Sch+13] und [MRD08]
Evaluation Handlungsoptionen	(+)	-	möglich durch Daten der Systeme von [Sch+13] und [MRD08]
Evaluation Prozessziele und -erfolg	+	+	durch Assertions auf Kontextpfad
Handlungsselektion			
durch Kosten und Risiken	(+)	-	möglich durch die Systeme von [Sch+13] und [MRD08]
durch Ereignisregeln, Zielfunktionen, Vorhersagemodelle	+	+	räumliche Lokalisation und Messgröße in Assertions und Kontextpfad
mit Regelkonfliktmanagement	+	+	manuell durch Modellierer
Lernen			

Anforderung	K	P	Erklärung
erfahrungsorientiert	-	-	fehlende Simulation, unverarbeitete Historie
inkrementelle Verbesserung	+	+	inkrementelle Minimierung des Mismatch
ermöglicht autonome Adaption	+	+	durch Rückkopplung über Zustand innerhalb der Selbstwahrnehmung
Reaktion			
durch Aktuatoren	+	+	Middleware (hier OpenHAB)
durch Anpassen von Param. des Netzwerks, der Performanz	(+)	-	möglich durch Middleware und Executer-Delegation
Management von Softwaremodulen	(+)	-	möglich durch Middleware und Executer-Delegation
kontrollierter Ressourcenzugriff	(+)	-	möglich durch Middleware und Executer-Delegation
Sicherheit			
betriebssicher	-	-	unbetrachtet
angriffssicher	-	-	unbetrachtet
Schutz der Privatsphäre	-	-	unbetrachtet
Validation/Verifikation von Daten	-	-	unbetrachtet

Zur besseren Übersicht werden die konzeptuellen und prototypisch erreichten Anforderungen in Abbildung 7.2 grafisch gegenübergestellt. Der jeweilige Wert beschreibt das Verhältnis zwischen abgedeckten und möglichen Anforderungsaspekten.

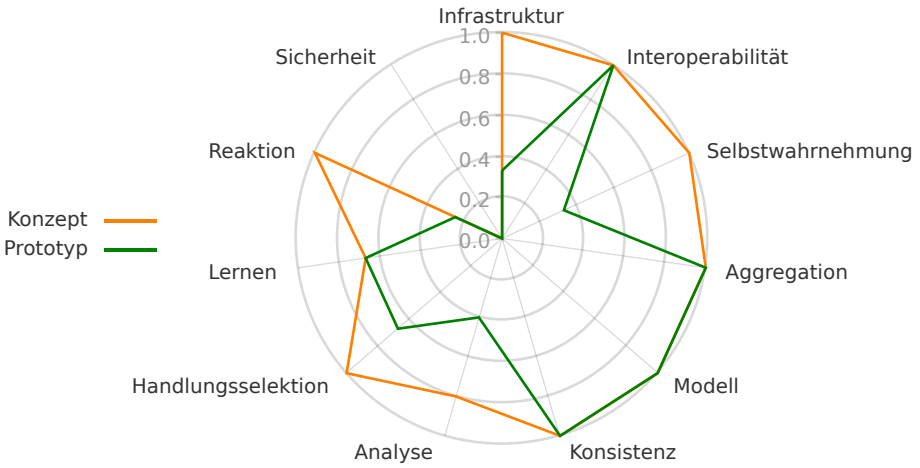

Abbildung 7.2: Anforderungsabdeckung bzgl. Konzept und Prototyp

7.3 Aufgetretene Probleme

Eine grundsätzliche Schwierigkeit bestand in der Vergleichbarkeit mit bestehenden Ansätzen. Jede betrachtete Arbeit beschäftigt sich mit Teilaspekten, kann die gesamtheitliche Integration von MAPE-K in CPS-Prozesse jedoch nicht bewerkstelligen. Außerdem bleibt die Sicherheit solcher Systeme weiterhin unbetrachtet, wodurch ein notwendiger Feldtest risikobehaftet und für beteiligte Personen nicht tragbar wäre.

Auf technischer Ebene ist vor allem die rein formale Beschreibung kontextsensitiver Ziele problematisch und kann mit diesem Konzept nur von Personen mit dementsprechenden System- und Kontextstruktur-Kenntnissen umgesetzt werden. Andererseits besteht in dem maschinenlesbaren Austauschformat die Möglichkeit, Goals und Objectives durch andere Subsysteme definieren zu lassen.

Die Integration in eine Prozessmodellierungssprache außerhalb von PROtEUS, wäre aufgrund der Umsetzung als ein mit Standards konformer Web-Service grundsätzlich möglich. Jedoch würden die Service-spezifischen Schritte mit der Geschäftslogik eines Prozesses vermengt, was die Übersichtlichkeit, den Modellierungsaufwand und die -wartung negativ beeinflusste.

In der Plan-Phase wird die Kompensation durch die Abbildung des Mismatch-Typen auf den Kommando-Typen identifiziert (vgl. Abschnitt 5.1.4). Dieses regelbasierte Verhalten ist schwer zu erweitern und sollte durch bessere Planungsmechanismen ersetzt werden. Ähnlich problematisch ist die Eindimensionalität der Soll- und Ist-Werte eines Mismatch. Zwar können die Messungen im Kontextpfad ver-

knüpft werden (z.B. `avg()`, vgl. Zeile 19, Listing A.1.3), jedoch ist der Vergleich mit mehrdimensionalen Sensoren kaum durch diese Art der Planung zu bewerkstelligen. Beispielsweise müsste für Koordinaten als Teil eines Kontext ein Mismatch-Typ wie `TOO_FAR` betrachtet werden, der einige konzeptuelle Änderungen nach sich ziehen würde.

7.4 Abschließende Betrachtungen

Der Fokus dieser Arbeit lag bei der Betrachtung von Verknüpfungsmöglichkeiten von MAPE-K und CPS-Prozessen. Neben der Eingrenzung unterschiedlichster Domain-Aspekte, nach umfangreichen Recherchen, und dem Gestalten einer konzeptuellen Lösung, ist der Prototyp Beweis der Funktion anhand einer Probleminstanz. Dieses Vorgehen ist teilweise der Design Science Research Methodology (DSRM) entlehnt, die eine Schablone für wissenschaftliches Arbeiten mit technischen Informationssystemen bietet und speziell auf konstruierende Forschung zugeschnitten wurde [Pef+08]. Um den Nutzen dieser Ausarbeitung zusammenfassend herauszustellen, ist in Tabelle 7.2 eine retrospektive DSRM-Analyse nach Geerts et al. nachzuvollziehen [Gee11]. Die erste Spalte listet jene Tätigkeiten, auf denen die DSRM beruht. In der zweiten sind Details der Aktivitäten beschrieben, die mit den Quellen der dritten Spalte verbunden werden.

Tabelle 7.2: Retrospektive DSRM-Analyse dieser Arbeit

Aktivität	Beschreibung	Informationsgrundlage
Problem-Identifikation und -Motivation	Das physikalische Verhalten von Prozessen in CPS muss Verifiziert und ggf. korrigiert werden, um die steigende Komplexität autonomer Systeme weiterhin beherrschen zu können.	Literaturrecherche zu selbstregulativen Systemen und Rückkopplung; Verständnis für Prozesse, CPS und MAPE-K; bestehende Ansätze auf Eignung prüfen

Aktivität	Beschreibung	Informationsgrundlage
Definition der Ziele einer Lösung	Anforderungen für die Integration von MAPE-K in CPS-Prozesse, -Subprozesse und einzelne Aktivitäten im Hinblick auf Modellierung und Ausführung, Simulation und Transaktionalität	Kombination von CPS- und MAPE-K-Anforderungen; etablierte Prozessmodellierungs- und Ausführungstechnologien; Lösungen für verteilte Transaktionen und BPS (vgl. Abschnitt 3, 2)
Design und Entwicklung	Konzeptuelles und prototypisches Design der in CPS-Prozesse eingebetteten Phasen von MAPE-K unter Berücksichtigung der Anforderungen (Exemplifikation nach [Pef+12]), vgl Abschnitt 5	Graphentheorie, Kontextmodellierungs- und Prozessdefinitionstechniken
Demonstration	Ein Prototyp kompensiert das unerfüllte Ziel des Szenarios, (vgl. Abschnitt 6, 7.1); technisches Experiment und Prototyp nach [Pef+12]	Kenntnis der Sprache zur Modellierung von Zielen und zum Festlegen der Assertions; Wissen um Struktur des Kontextmodells; Infrastruktur; PROtEUS- und OpenHAB-Dokumentation
Evaluation	Erfolgreiche Kompensation der Probleminstanz; Anforderungsabgleich bzgl. Konzept und Prototyp (vgl. Abschnitt 7)	Test des Prototypen unter den künstlichen Bedingungen des Szenarios (vgl. Abschnitt 7.1)

Aktivität	Beschreibung	Informationsgrundlage
Kommunikation	Verteidigung vor dem verantwortlichen, Hochschullehrer, den Betreuern der Arbeit, einem Auditorium von Promovenden/Interessierten	Abschlusspräsentation, schriftliche Ausarbeitung, Quellcode des Prototypen

Forschungskontext. Während der Recherche zu den Dimensionen dieser Arbeit, konnte kein rückgekoppeltes, autonomes Gesamtsystem, mit dem Schwerpunkt auf Prozessen, gefunden werden. Viele der Erkenntnisse boten Lösungen für Teilepro-bleme eines solchen, konnten aber bezüglich ihrer Verknüpfung nicht betrachtet werden, ohne den Fokus zu verlieren. Somit bietet dieses Konzept ein initiales Modell für die Integration von MAPE-K in CPS-Prozesse.

Allgemeingültigkeit. Die Übertragbarkeit des Konzepts auf Anwendungsfälle außerhalb des Laborexperiments, wäre mit einem Feldexperiment zu verifizieren. Innerhalb dessen ist ein realitätsnaher Kontext entweder mit genügend Ressourcen oder Simulationsmechanismen bereitzustellen. Mit ausreichend Zeit und Messungen, wären folgende beispielhafte Metriken interessant für die Evaluation der Leistung des Systems.

- Das Verhältnis der erfolgreichen und erfolglosen Prozesse, mit und ohne kompensierende Rückkopplung.
- Die zeitliche Verzögerung des Prozesserfolgs durch MAPE-K.
- Die Anzahl der notwendigen Schleifeniterationen.

8 Zusammenfassung und Ausblick

In diesem letzten Kapitel wird ein abschließendes Fazit gezogen und eine Menge an mögliche Erweiterungen vorgestellt.

8.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde die Integration der MAPE-K-Rückkopplungsschleife in CPS-Prozesslandschaften betrachtet. Die Kombination der Anforderungen an derlei Systeme, bildete die Grundlage der Bewertung bestehender Forschungsergebnisse. In Anlehnung an die Erkenntnisse vorangegangener Recherchen, wurde ein Konzept entworfen, ein vertikaler Prototyp entwickelt und ein technisches Experiment zur Evaluation durchgeführt. Dabei traten folgende Ergebnisse in den Vordergrund.

1. Rückkopplung mit MAPE-K kann zu einer besseren Prozess-Erfolgsquote in CPS führen. Wo einzelne Aktivitäten scheitern, kann die Schleife eingreifen um das ursprüngliche Ziel zu gewährleisten. Im Kontext dieser Arbeit wurde die Kompensation von erfolglosen Prozessen durch Aktuatoren bewerkstelligt.
2. Für den Einsatz von MAPE-K, müssen die Ziele eines Prozesses, Subprozesses oder die einer Aktivität explizit unter Kenntnis der Kontextstruktur formuliert, beziehungsweise modelliert werden. Eine implizite, transparente Integration der Kontrollschleife in einen CPS-Prozess ist mit diesem Ansatz nicht möglich.
3. Eine generische Integration der Rückkopplung in beliebige Prozessmodellierungssprachen und Ausführungsumgebungen ist ohne Anpassungen möglich. Service-Aufrufe durch REST, SOAP und ähnliche Protokolle, erlauben die Anbindung an eine eigenständige MAPE-K-Komponente. Das gezielte Einbetten in PROtEUS wurde durch das Einbinden eines REST-Web-Service bewerkstelligt.
4. Die Anbindung an bestehende Sensor-/Aktuator-Systeme und Middlewares, ist gekapselt und kann problemlos erweitert werden.

5. Prozesse können mit MAPE-K implizit transaktionalisiert werden. Kompensation und Wiederholbarkeit sind, als Teil von Semiatomizität, in die Planungsphase der Rückkopplung eingeflossen.
6. Durch die Echtzeit- und Isolationsanforderung an einen rückgekoppelten Prozess, konnten Simulationsprinzipien nicht in den operativen Betrieb integriert werden. Die Laufzeitadaption beschränkt sich hier auf regelbasiertes Verhalten.
7. In dieser Arbeit, bleiben Betriebs- und Angriffssicherheit im Kontext von CPS und IoT unbetrachtet.

Insgesamt wurde festgestellt, dass eine vollständige Selbstregulation von Prozessen, die physikalische Parameter manipulieren, durch dieses Konzept begründet, jedoch nicht in Gänze geleistet werden kann. Somit beschreibt diese Arbeit nicht nur die erfolgreiche Verbindung von CPS-Prozessen mit der MAPE-K-Rückkopplungsschleife, sondern bietet außerdem neues Material für zahlreiche weiterführende Forschungsfragen.

8.2 Ausblick

Reaktive Planungsphase. Statisches, regelbasiertes Planen ist aufwendig und inflexibel. Bereits prototypisch implementiert, kann die Historie der Kontextveränderungen in der Planungsphase genutzt werden. Unter den geeigneten Methoden für die inkrementelle Verbesserung der Kontrollschleife finden sich Neuronale Netze, Stützvektormaschinen, Agenten-basierte Ansätze und fallbasiertes Schließen. Weiterhin ist eine kompensatorische Lösungsfindung über das Aufspannen eines Zustandsraums und entsprechende Suchalgorithmen möglich.

Rückkopplung und Business-Logik. Um die in Prozesse gegossene Geschäftslogik nicht mit expliziter Rückkopplungsinformation zu belasten, sollte eine bessere Trennung untersucht werden. Eine Transformation des Prozessmodells über sein Austauschformat (meist XML) kann den eigentlichen Prozess als Subprozess eines durch MAPE-K überwachten abbilden. Das Einbetten der Zielinformationen muss dann auf einem noch zu untersuchenden Weg geschehen.

Erweiterung der DogOnt-Ontologie. Über die semantische Verknüpfung eines Sensors mit dessen Aktuator-Pendant, kann die DogOnt-Ontologie für eine bessere Planung eingesetzt werden. Die komplexe, nicht ausreichend evaluierte Methode

des Findens einer Kompensation kann so wesentlich einfacher gestaltet werden. Im Zuge dessen, ist auch das Problem der Skalarität von Mismatches von Bedeutung. Denkbar ist hier das Erlernen der Verbindungen zwischen Sensoren und Aktuatoren.

Kontextsensitive Prozessziele. Die ProGoalML liefert eine durchdachte Methode zur Formalisierung von Zielen und zur Integration in die Prozessausführungslandschaft [KK12]. Eine Untersuchung möglicher kontextsensitiver Zieldefinitionen mit dieser Sprache kann das Rückkopplungsproblem entlasten und Interoperabilität verbessern.

Verbesserung der Selbstwahrnehmung. Der Web-Service selbst kann das managed Element einer MAPE-K-Schleife sein. Metriken wie die Serverlast und Netzwerkqualität wären dann Messgrößen im Kontext des Monitorings. Rückkopplung kann somit die Selbstwahrnehmung des Systems auf einem höheren Abstraktionsniveau bewerkstelligen und beispielsweise horizontale Skalierung für verteilte Prozesse vereinfachen. Ein Cloud-basierter Ansatz oder die Platform as a Service werden bereits produktiv eingesetzt und können mit MAPE-K verbunden werden (vgl. ViePEP, VieDAME, Abschnitt 3.1).

Transaktionalität durch WS-TX. Die Untersuchung der Verbindung des in dieser Arbeit vorgestellten Konzepts mit den WS-TX-Protokollen, kann den Aspekt der Transaktionalität von CPS-Prozessen erweitern (vgl. Abschnitt 5.3). Aktuell geht dieser nicht über die Kompensation fehlgeschlagener Ziele hinaus.

A Anhang

A.1 Prototypischer Web-Service

A.1.1 Installation

Die Anwendung verwendet Gradle[1] für den Build-Prozess und Spring Boot für die Laufzeit. Für Windows existiert eine Batch- (`gradlew.bat`) und für Unix eine Bash-Datei (`./gradlew`), im folgenden als `gradle` bezeichnet. Der Service kann mit `gradle bootRun` direkt gestartet werden. Parameter aus Abschnitt A.1.2 werden durch `-D` übergeben (z.B. `gradle bootRun -Dserver.port=9001`). Weiter mögliche Wege der Übergabe sind in der Dokumentation von Spring Boot nachzulesen[2].

A.1.2 Konfiguration

Die folgende Tabellen umfasst die wichtigsten Parameter des Prototypischen Web-Services.

Tabelle A.1: Parameter und deren Beschreibung

Parameter	Beschreibung
openHab.host	OpenHAB-Hostname oder -IP
openHab.port	OpenHAB-Port
openHab.delta	Schwellwert für Filter (vgl. Abschnitt 6.3.2)

[1] www.gradle.org
[2] docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html

Parameter	Beschreibung
openHab.pollingSeconds	Zeit zw. den Aktualisierungen der Items
dogOnt.thingNodePrefix	Prefix des Namens eines Thing-knotens
dogOnt.stateNodePrefix	Prefix des Namens eines Zustandsknotens
server.port	Web-Service Port
spring.profiles.active	Profilliste (keine Leerzeichen)
logging.level.de.tud	Grad des Anwendungslogs
service.knowledge.neo4j.url	Adresse zu Neo4j-Instanz
service.knowledge.neo4j.username	Nutzername für Neo4j
service.knowledge.neo4j.password	Passwort für Neo4j

Tabelle A.2: Parameter und deren Wertebereiche

Parameter	Wertebereich
openHab.host	Hostname, IP
openHab.port	Port
openHab.delta	[0, 1]
openHab.pollingSeconds	ab 0 Sekunden
dogOnt.thingNodePrefix	String
dogOnt.stateNodePrefix	String
server.port	Port

Parameter	Wertebereich
spring.profiles.active	development, embeddedNeo4j
logging.level.de.tud	TRACE, DEBUG, INFO, WARN, ERROR
service.knowledge.neo4j.url	URL
service.knowledge.neo4j.username	String
service.knowledge.neo4j.password	String

Tabelle A.3: Parameter und deren Standardwerte

Parameter	Standardwert
openHab.host	localhost
openHab.port	8080
openHab.delta	0,01
openHab.pollingSeconds	1
dogOnt.thingNodePrefix	Thing_
dogOnt.stateNodePrefix	State_
server.port	9000
spring.profiles.active	development,embeddedNeo4j
logging.level.de.tud	INFO
service.knowledge.neo4j.url	http://localhost:7474/
service.knowledge.neo4j.username	

Parameter	Standardwert
service.knowledge.neo4j.password	

A.1.3 Benutzung

Erzeugen des Kontexts durch eine OpenHAB-Instanz auf dem lokalen System.

```
POST /contexts HTTP/1.1
Host: localhost:9000
Content-Type: application/json

{
    "name": "home",
    "imports": [{
        "source": "classpath:dogont.owl",
        "mimeType": "application/rdf+xml",
        "name": "ontology"
    },{
        "source": "http://localhost:8080/rest/semantic",
        "mimeType": "application/rdf+xml",
        "name": "items"
    }]
}
```

Anlegen einer Process-Componente (`Workflow`) mit der Referenz auf den zuvor erzeugten Kontext.

```
POST /workflows HTTP/1.1
Host: localhost
Content-Type: application/json

{
  "name": "prepare cooking with friends",
  "context": "http://localhost/contexts/3132"
}
```

Festlegen von Goals und deren Objectives, relativ zur Process-Component.

```
POST /goals HTTP/1.1
Host: localhost
Content-Type: application/json

{
```

```
6     "name": "enough light for cooking",
7     "workflow": "http://localhost/workflows/3133",
8     "objectives": [{
9       "name": "kitchen light > 865 lux within two seconds",
10      "satisfiedExpression": "#lighting > 865",
11      "compensateExpression":
12          "#objective.created.isBefore(#now.minusSeconds(2))",
13      "testNodeIdExpression": "#stateId",
14      "contextExpressions": [
15        "MATCH (t)-[:type]->(s { name: 'LightSensor' })",
16        "MATCH (t)-[:isIn]->(r { name: 'Küche' })",
17        "MATCH (t)-[:hasState]->(s:LightIntensityState)",
18        "MATCH (s)-[:hasStateValue]->(v)",
19        "RETURN avg(toFloat(v.realStateValue)) AS lighting, ",
20                          "head(collect(id(s))) AS stateId"
21      ]
22    }]
23  }
```

Verbindung via SSE zu */events/workflows/3133* herstellen. Sobald die Kontrollschleife fertiggestellt ist, wird hier ein JSON-serialisiertes `Workflow`-Objekt zurückgegeben. Abschließend sollte die Process-Component gelöscht werden (HTTP-DELETE), um keine verwaisten Knoten zu hinterlassen.

A.1.4 Entwicklung

A.1.4.1 IntelliJ

1. *File* > *New* > *Project from existing Sources*
2. Select *build.gradle* from the project's root

A.1.4.2 Eclipse

1. Execute `gradle eclipse`
2. *File* > *Import* > *Existing Projects into Workspace*
3. Choose the project's root directory
4. Check *Search for nested Projects*

Akronyme

2PC Two-Phase-Commit.

AAL Ambient Assisted Living.
ABIS Adaptive Business Process Modeling in the Internet of Services.
ACID Atomicity, Consistency, Isolation, Durability.
AOP aspektorientierte Programmierung.
AP Application Program.
API Advanced Programming Interface.
aPro Architecture for Business Process optimization.
AST Abstract Syntax Tree.

BAC Business Agreement with Coordinator Completion Protocol.
BAM Business Activity Monitoring.
BAP Business Agreement with Participation Completion Protocol.
BPM Business Process Management.
BPMN Business Process Model and Notation.
BPMS Business Process Management Systems.
BPS Business Process Simulation.

CCC Crosscutting Concerns.
CEP Complex Event Processing.
CORBA Common Object Request Broker Architecture.
CPS cyber-physische Systeme.
CRUD Create, Read, Update, Delete.

DES Discrete-Event Simulation.
DESMO Discrete-Event Simulation and Modeling.
DISMO Distributed Simulation Optimization.
DSRM Design Science Research Methodology.

EDBPM Event-Driven Business Process Management.
EJB Enterprise JavaBeans.
EMF Eclipse Modeling Framework.
EPA Event-Processing Agent.
EPL Event-Processing Language.

EPN Event-Processing Network.

GA genetischer Algorithmus.
GUI Graphical User Interface.

IoT Internet of Things.

JML Java Modeling Language.
JSON JavaScript Object Notation.

KPI Key Performance Indicator.

MAPE-K Monitor, Analyze, Plan, Execute und Knowledge.
MOF Meta Object Facility.

OASIS Advancing open standards for the information society.
OMG Object Management Group.
OO Objekt-Orientierung.
OSGi Open Services Gateway initiative.
OTS Object Transaction Service.
OWL Web Ontology Language.

PAIS Process-Aware Information System.
PC Process-Component.
PE Prozess-Engine.
POJO Plain old Java-Object.
ProGoalML Process Goal Markup Language.
PROtEUS Process Execution Environment for Ubiquitous Systems.

QoS Quality of Service.

RDF Resource Description Framework.
REST Representational State Transfer.
RFID Radio-Frequency Identification.
RM Resource Manager.
RMI Remote Method Invocation.

SH Smart Home.
SOAP Simple Object Access Protocoll.
SoC Separation of Concerns.
SpEL Spring Expression Language.
SPOF Single Point of Failure.
SQL Structured Query Language.
SRP Single-Responsibility-Prinzip.
SSE Server-Sent Events.

TM Transaction Manager.

UML Unified Modeling Language.
URI Uniform Resource Identifier.
URL Uniform Resource Locator.

VieDAME Vienna Dynamic Adaption and Monitoring Enviroment for WS-BPEL.
ViePEP Vienna Platform for Elastic Processes.

WB Wissensbasis.
WfMS Workflow Management System.
WS-BPEL Web-Service Business Process Execution Language.
WS-CoL Web-Service Constraint Language.
WS-TX Web Services Transaction.

XA eXtended Architecture.
XML Extensible Markup Language.
XML-RPC XML Remote Procedure Call.
XSD XML Schema Definition.

x

Abbildungen

Programmcode

Tabellenverzeichnis

Literatur

[Aal15] Wil M. P. van der Aalst. „Business Process Simulation Survival Guide“. In: *Handbook on Business Process Management 1*. 2015, Seiten 337–370. DOI: 10.1007/978-3-642-45100-3_15 (siehe Seiten 27, 28).

[AHW03] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede und Mathias Weske. „Business Process Management: A Survey“. In: *Business Process Management*. Springer Berlin Heidelberg, 2003, Seiten 1–12. DOI: 10.1007/3-540-44895-0_1 (siehe Seiten 6, 7, 15).

[AIM10] Luigi Atzori, Antonio Iera und Giacomo Morabito. „The Internet of Things: A survey“. In: *Computer Networks* 54.15 (Okt. 2010), Seiten 2787–2805. DOI: 10.1016/j.comnet.2010.05.010 (siehe Seiten 3, 4).

[Alo05] Gustavo Alonso. „Transactional Business Processes“. In: *Process-Aware Information Systems: Bridging People and Software through Process Technology* (2005), Seiten 257–278. DOI: 10.1002/0471741442.ch11 (siehe Seiten 23, 25, 26).

[Ash09] Kevin Ashton. *That 'Internet of Things' Thing*. 2009. URL: http://www.rfidjournal.com/articles/view?4986 (besucht am 11. 06. 2015) (siehe Seite 3).

[AZ06] Uwe Aßmann und Steffen Zschaler. „Ontologies, Meta-models, and the Model- Driven Paradigm“. In: *Ontologies for Software Engineering and Software Technology*. Springer Berlin Heidelberg, 2006, Seiten 249–273. DOI: 10.1007/3-540-34518-3_9 (siehe Seite 8).

[BCG12] Manfred Broy, María Victoria Cengarle und Eva Geisberger. „Cyber-physical systems: Imminent challenges“. In: *Large-Scale Complex IT Systems. Development, Operation and Management*. Band 7539 LNCS. 2012, Seiten 1–28. DOI: 10.1007/978-3-642-34059-8_1 (siehe Seiten 31–34, 36–41).

[BG05] Luciano Baresi und Sam Guinea. „Towards Dynamic Monitoring of WS-BPEL Processes“. In: *Service-Oriented Computing - ICSOC 2005*. Band 3826. Springer Berlin Heidelberg, 2005, Seiten 269–282. DOI: 10.1007/11596141_21 (siehe Seiten 16, 32, 34, 37, 40, 41).

[Bru+09] Yuriy Brun, Giovanna Marzo Serugendo, Cristina Gacek u. a. „Engineering Self-Adaptive Systems Through Feedback Loops“. In: *Software Engineering for Self-Adaptive Systems*. Springer, 2009, Seiten 48–70. DOI: 10.1007/978-3-642-02161-9_3 (siehe Seiten 10, 12, 13).

[CAB08] CABA. *Internet of Things in 2020 - Roadmap for the Future (IS-2008-93)*. Technischer Bericht. 2008, Seite 29. URL: http://www.caba.org/DocumentHandler.ashx?DocId=18896 (siehe Seite 4).

[CB08] Fulvio Corno und Dario Bonino. „DogOnt - Ontology Modeling for Intelligent Domotic Environments“. In: *ISWC '08 - Proceedings of the 7th International Conference on The Semantic Web*. 2008, Seiten 790–803. DOI: 10.1007/978-3-540-88564-1_51 (siehe Seiten 45, 52).

[Che10] Guihai Chen. „Internet of Things Towards Ubiquitous and Mobile Computing“. In: *Microsoft Research Asia Faculty Summit 2010*. Shanghai, 2010. URL: http://research.microsoft.com/en-us/UM/redmond/events/asiafacsum2010/presentations/Guihai-Chen_Oct19.pdf (siehe Seite 5).

[Dau+13] Rustem Dautov, Dimitrios Kourtesis, Iraklis Paraskakis u. a. „Addressing self-management in cloud platforms“. In: *International workshop on Hot topics in cloud services*. ACM, 2013, Seite 11. DOI: 10.1145/2462307.2462312 (siehe Seite 13).

[DAV12] Massimiliano De Leoni, Wil M. P. van der Aalst und Boudewijn F. Van Dongen. „Data- and resource-aware conformance checking of business processes“. In: *Lecture Notes in Business Information Processing* (2012), Seiten 48–59. DOI: 10.1007/978-3-642-30359-3_5 (siehe Seiten 17, 19, 35).

[DK75] Frank DeRemer und Hans Kron. „Programming-in-the large versus programming-in-the-small“. In: *Proceedings of the international conference on Reliable software*. Band 10. 6. ACM, 1975, Seiten 114–121. DOI: 10.1145/800027.808431 (siehe Seite 6).

[Fow03] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003. ISBN: 9780321127426 (siehe Seite 23).

[Gar+12] Marcela Garcia, Kelly Braghetto, Calton Pu u. a. „An implementation of a transaction model for business process systems“. In: *Journal of Information and Data Management* 3.3 (2012), Seite 271. URL: https://seer.lcc.ufmg.br/index.php/jidm/article/view/198 (siehe Seite 24).

[Gee11] Guido L. Geerts. „A design science research methodology and its application to accounting information systems research“. In: *International Journal of Accounting Information Systems* 12.2 (Juni 2011), Seiten 142–151. DOI: 10.1016/j.accinf.2011.02.004 (siehe Seite 77).

[GP01] Björn Gehlsen und Bernd Page. „A framework for distributed simulation optimization“. In: *Simulation Conference, 2001. Proceedings of the Winter* 1 (2001), Seiten 508–514. DOI: 10.1109/WSC.2001.977331 (siehe Seiten 30, 38–41).

[GS87] Hector Garcia-Molina und Kenneth Salem. „Sagas“. In: *Proceedings of the 1987 ACM SIGMOD*. ACM, 1987, Seiten 249–259. DOI: 10.1145/38713.38742 (siehe Seiten 24, 36).

[Gur+13] Levent Gurgen, Ozan Gunalp, Yazid Benazzouz u. a. „Self-aware cyber-physical systems and applications in smart buildings and cities“. In: *DATE '13 Proceedings of the Conference on Design, Automation and Test in Europe* (2013), Seiten 1149–1154. DOI: 10.7873/DATE.2013.240 (siehe Seiten 31–33, 35–41).

[HBM08] Rachid Hamadi, Boualem Benatallah und Brahim Medjahed. „Self-adapting recovery nets for policy-driven exception handling in business processes“. In: *Distributed and Parallel Databases* 23.1 (2008), Seiten 1–44. DOI: 10.1007/s10619-007-7020-1 (siehe Seite 24).

[Hoe+13] Philipp Hoenisch, Stefan Schulte, Schahram Dustdar u. a. „Self-adaptive resource allocation for elastic process execution“. In: *6th IEEE International Conference on Cloud Computing* (2013), Seiten 220–227. DOI: 10.1109/CLOUD.2013.126 (siehe Seite 15).

[HSD10] Gabriel Hermosillo, Lionel Seinturier und Laurence Duchien. „Using Complex Event Processing for Dynamic Business Process Adaptation“. In: *Proceedings of the 7th international Conference on Services Computing*. IEEE, 2010, Seiten 466–473. DOI: 10.1109/SCC.2010.48 (siehe Seite 20).

[IBM06] IBM. *An architectural blueprint for autonomic computing*. Technischer Bericht June. 2006. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.150.1011 (siehe Seiten 11–13).

[JMM12] Christian Janiesch, Martin Matzner und Oliver Müller. „Beyond process monitoring: a proof-of-concept of event-driven business activity management“. In: *Business Process Management Journal* 18.4 (2012), Seiten 625–643. DOI: 10.1108/14637151211253765 (siehe Seiten 15, 20, 32, 33, 35, 37, 38, 40, 41).

[KH01] Gregor Kiczales und Erik Hilsdale. „Aspect-oriented programming“. In: *Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering - ESEC/FSE-9*. Band 26. 5. ACM, 2001, Seite 313. DOI: 10.1145/503209.503260 (siehe Seite 16).

[KK12] Falko Koetter und Monika Kochanowski. „Goal-Oriented Model-Driven Business Process Monitoring Using ProGoalML“. In: *Business Information Systems*. Springer, 2012, Seiten 72–83. DOI: 10.1007/978-3-642-30359-3_7 (siehe Seiten 21, 22, 32, 35, 37–41, 45, 83).

[KK13] Falko Koetter und Monika Kochanowski. „A model-driven approach for event-based business process monitoring“. In: *Business Process Management Workshops*. Springer Berlin Heidelberg, 2013, Seiten 378–389. DOI: 10.1007/978-3-642-36285-9_41 (siehe Seiten 32, 35, 41).

[Küh15] André Kühnert. „Konzeption eines Semantischen Access-Layers für die OpenHAB- Plattform“. Bachelorarbeit. Technische Universität Dresden, 2015 (siehe Seite 61).

[LK06] Beate List und Birgit Korherr. „An Evaluation of Conceptual Business Process Modelling Languages“. In: *ACM symposium on Applied computing*. ACM, 2006, Seiten 1532–1539. DOI: 10.1145/1141277.1141633 (siehe Seite 17).

[LMM15] Francesco Leotta, Massimo Mecella und Jan Mendling. „Applying Process Mining to Smart Spaces: Perspectives and Research Challenges“. In: *Advanced Information Systems Engineering Workshops*. Band 215. Springer, 2015, Seiten 298–304. DOI: 10.1007/978-3-319-19243-7_28 (siehe Seite 17).

[Mat08] Norm Matloff. *Introduction to discrete-event simulation and the simpy language*. 2008. URL: http://heather.cs.ucdavis.edu/%7B~%7Dmatloff/156/PLN/DESimIntro.pdf (siehe Seite 28).

[MC10] Jorge Muñoz-Gama und Josep Carmona. „A fresh look at precision in process conformance“. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6336 LNCS (2010), Seiten 211–226. DOI: 10.1007/978-3-642-15618-2_16 (siehe Seiten 18, 19, 35).

[MRD08] Oliver Moser, Florian Rosenberg und Schahram Dustdar. „VieDAME - flexible and robust BPEL processes through monitoring and adaptation“. In: *30th International Conference on Software Engineering* August 2015 (2008), Seiten 917–918. DOI: 10.1145/1370175.1370186 (siehe Seiten 16, 32–34, 37–39, 74).

[OAS07] OASIS. *Web Services Business Process Execution Language (WS-BPEL)*. 2007. URL: http://docs.oasis-open.org/wsbpel/2.0/ (besucht am 05. 08. 2015) (siehe Seite 7).

[OAS09] OASIS. *Web Services Transaction (WS-TX)*. 2009. URL: http://docs.oasis-open.org/ws-tx/ (besucht am 05. 08. 2015) (siehe Seiten 26, 32, 33, 36–38, 40, 41).

[Obj11] Object Management Group (OMG). *Business Process Model and Notation (BPMN) 2.0*. 2011. URL: http://www.omg.org/spec/BPMN/2.0/ (besucht am 05. 08. 2015) (siehe Seite 7).

[Obj15] Object Management Group (OMG). *OMG Meta Object Facility (MOF) Core Specification 2.5*. 2015. URL: http://www.omg.org/spec/MOF/2.5/ (besucht am 05. 08. 2015) (siehe Seite 8).

[Ope92] Open Group. *XA Specification*. 1992. URL: https://www2.opengroup.org/ogsys/catalog/c193 (besucht am 05. 08. 2015) (siehe Seiten 25, 32, 33, 36–38, 40, 41).

[Pef+08] Ken Peffers, Tuure Tuunanen, Marcus a. Rothenberger u. a. „A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3 (2008), Seiten 45–77. DOI: 10.2753/MIS0742-1222240302 (siehe Seite 77).

[Pef+12] Ken Peffers, Marcus Rothenberger, Tuure Tuunanen u. a. „Design Science Research Evaluation". In: *Design Science Research in Information Systems. Advances in Theory and Practice* (2012), Seiten 398–410. ISSN: 0302-9743. DOI: 10.1007/978-3-642-29863-9_29 (siehe Seite 78).

[Por+06] Alberto Portilla, Genoveva Vargas-Solar, José Luis Zechinelli-Martini u. a. „A survey for analyzing transactional behavior in service based applications". In: *Proceedings of the Mexican International Conference on Computer Science*. IEEE, 2006, Seiten 116–123. DOI: 10.1109/ENC.2006.2 (siehe Seite 24).

[RA08] Anne Rozinat und Wil M. P. van der Aalst. „Conformance checking of processes based on monitoring real behavior". In: *Information Systems* 33.1 (2008), Seiten 64–95. DOI: 10.1016/j.is.2007.07.001 (siehe Seiten 17, 18, 35).

[Raj+10] Ragunatham Rajkumar, Insup Lee, Lui Sha u. a. „Cyber-physical systems: The next computing revolution". In: *DAC '10 Proceedings of the 47th Design Automation Conference*. ACM, 2010, Seiten 731–736. DOI: 10.1145/1837274.1837461 (siehe Seite 5).

[Rüc08] Bernd Rücker. „Building an open source Business Process Simulation tool with JBoss jBPM". Master Thesis. Stuttgart University of applied science, 2008 (siehe Seiten 27, 28, 32, 37–40).

[Rus13] Siegfried Russwurm. „Software: Die Zukunft der Industrie". In: *Industrie 4.0*. Band 19. 1. Springer, 2013, Seiten 21–36. DOI: 10.1007/978-3-642-36917-9_2 (siehe Seite 4).

[Sch+13] Stefan Schulte, Philipp Hoenisch, Srikumar Venugopal u. a. „Introducing the vienna platform for elastic processes". In: *Service-Oriented Computing - ICSOC 2012 Workshops*. Springer, 2013, Seiten 179–190. DOI: 10.1007/978-3-642-37804-1_19 (siehe Seiten 15, 32, 34, 35, 37–41, 74).

[Sch09] Heiko Schuldt. „Transactional Processes“. In: *Encyclopedia of Database Systems*. Springer, 2009, Seiten 3166–3166. DOI: 10.1007/978-0-387-39940-9_734 (siehe Seite 24).

[Sei+13] Ronny Seiger, Christine Keller, Florian Niebling u. a. „Modelling complex and flexible processes for smart cyberphysical environments“. In: *25th European Modeling and Simulation Symposium, EMSS 2013* (Aug. 2013), Seiten 73–82. DOI: 10.1016/j.jocs.2014.07.001 (siehe Seiten 6–9, 11, 32, 44).

[Sha98] Robert E. Shannon. „Introduction to the art and science of simulation“. In: *Proceedings of the 1998 Winter Simulation Conference* 1 (1998), Seiten 7–14. DOI: 10.1109/WSC.1998.744892 (siehe Seite 27).

[SHS15] Ronny Seiger, Steffen Huber und Thomas Schlegel. „PROtEUS: An Integrated System for Process Execution in Cyber-Physical Systems“. In: *Enterprise, Business-Process and Information Systems Modeling*. 214. Springer, 2015, Seiten 265–280. DOI: 10.1007/978-3-319-19237-6_17 (siehe Seiten 8, 59).

[SS12] Erwin Schoitsch und Amund Skavhaug. „Introduction to the ERCIM/E-WICS Cyberphysical Systems Workshop 2012“. In: *Computer Safety, Reliability, and Security*. Springer, 2012, Seiten 343–346. DOI: 10.1007/978-3-642-33675-1_30 (siehe Seite 5).

[Sta73] Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer, 1973. ISBN: 9783211811061. URL: http://d-nb.info/101845750X (siehe Seite 8).

[Vid+15] Carlos Vidal, Carlos Fernández-Sánchez, Jessica Díaz u. a. „A Model-Driven Engineering Process for Autonomic Sensor-Actuator Networks“. In: *International Journal of Distributed Sensor Networks* (2015), Seiten 1–13. DOI: 10.1155/2015/684892 (siehe Seiten 11, 13).

[Wei91] Mark Weiser. „The computer for the 21st century“. In: *Scientific American* 265.3 (1991), Seiten 94–104. DOI: 10.1145/329124.329126 (siehe Seite 3).

[Zha+94] Aidong Zhang, Marian Nodine, Bharat Bhargava u. a. „Ensuring relaxed atomicity for flexible transactions in multidatabase systems“. In: *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*. Band 23. 2. ACM, 1994, Seiten 67–78. DOI: 10.1145/191839.191850 (siehe Seiten 25, 36).