Fraunhofer
IAO

FRAUNHOFER-INSTITUT FÜR ARBEITSWIRTSCHAFT UND ORGANISATION IAO

BIEC-SCHRIFTENREIHE ZUR DIGITALEN TRANSFORMATION IN KMU NR. 3
WILHELM BAUER, OLIVER RIEDEL, JENS NEUHÜTTLER, ANETTE WEISBECKER (HRSG.)

ÜBERWACHUNG UND STÖRUNGSERKENNUNG IN VERNETZTEN IT-SYSTEMEN

AUTOREN: FALKO KÖTTER | ANDREAS FREYMANN | MAXIMILIEN KINTZ | KRISTIAN SCHAEFER | THOMAS RENNER

FRAUNHOFER VERLAG

Fraunhofer
IAO

FRAUNHOFER-INSTITUT FÜR ARBEITSWIRTSCHAFT UND ORGANISATION IAO

ÜBERWACHUNG UND STÖRUNGSERKENNUNG IN VERNETZTEN IT-SYSTEMEN

Falko Kötter, Andreas Freymann, Maximilien Kintz, Kristian Schaefer, Thomas Renner
Herausgeber: Wilhelm Bauer, Oliver Riedel, Jens Neuhüttler, Anette Weisbecker

Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO
in Stuttgart

FRAUNHOFER VERLAG

Impressum

Kontaktadresse:
Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO)
Nobelstraße 12
70569 Stuttgart
Telefon +49 711 970-5120
Telefax +49 711 970-5111
E-Mail falko.koetter@iao.fraunhofer.de
URL http://www.iao.fraunhofer.de

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.de abrufbar.

ISBN (Print) 978-3-8396-1646-8

Druck und Weiterverarbeitung:
RCOM Print GmbH, Würzburg

Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

© FRAUNHOFER VERLAG, 2020
Fraunhofer-Informationszentrum Raum und Bau IRB
Postfach 800469, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 0711 970-2500
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

Alle Rechte vorbehalten

Dieses Werk ist einschließlich aller seiner Teile urheberrechtlich geschützt. Jede Verwertung, die über die engen Grenzen des Urheberrechtsgesetzes hinausgeht, ist ohne schriftliche Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikro-verfilmungen sowie die Speicherung in elektronischen Systemen.
Die Wiedergabe von Warenbezeichnungen und Handelsnamen in diesem Buch berechtigt nicht zu der Annahme, dass solche Bezeichnungen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und deshalb von jedermann benutzt werden dürften.
Soweit in diesem Werk direkt oder indirekt auf Gesetze, Vorschriften oder Richtlinien (z.B. DIN, VDI) Bezug genommen oder aus ihnen zitiert worden ist, kann der Verlag keine Gewähr für Richtigkeit, Vollständigkeit oder Aktualität übernehmen.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

1 Vorwort

Die Bedeutung von vernetzten IT-Systemen ist in den letzten Jahren in vielen Unternehmen im Zuge der digitalen Transformation stark gestiegen. Durch die wachsende Verbreitung von Cloud-Technologien, Anwendungen der Künstlichen Intelligenz sowie dem Internet der Dinge und den daraus resultierenden Anforderungen an IT-Systeme, wird sich diese Entwicklung auch künftig fortsetzen. Ungeachtet ihrer zahlreichen Vorteile, weisen vernetzte und insbesondere verteilte IT-Systeme eine hohe Komplexität auf, welche es durch den Einsatz von Instrumenten zur Überwachung und Störungserkennung zu beherrschen gilt. Neben traditionellen Werkzeugen, welche in verteilten Systemen zunehmend an ihre Grenzen stoßen, wurden in den vergangenen Jahren auch viele neue IT-Lösungen entwickelt. Die Auswahl geeigneter Instrumente für den jeweiligen Anwendungsfall stellt dabei eine große Herausforderung dar.

Die vorliegende Marktstudie ist Teil der Schriftenreihe »Digitale Transformation in KMU« des Business Innovation Engineering Centers (BIEC) und soll Unternehmen bei der Auswahl geeigneter Werkzeuge zur Überwachung und Störungserkennung in verteilten Systemen unterstützen. Ziel dabei ist es, den sicheren und zuverlässigen Betrieb dieser Systeme insbesondere in kleinen und mittleren Unternehmen (KMU) zu fördern und somit ihre Komplexität beherrschbar zu machen. Neben einer Marktuntersuchung wurden dafür 28 Experten befragt. Als Ergebnis zeigt die Studie Handlungsoptionen auf, stellt aktuell am Markt verfügbare technische Lösungen systematisch vor und beschreibt praktische Erfahrungen aus der Anbieter- und der Anwenderperspektive.

Das BIEC wird vom Ministerium für Wirtschaft, Arbeit und Wohnungsbau Baden-Württemberg gefördert und bietet umsetzungsorientierte Entwicklungs- und Transfermaßnahmen mit dem Ziel an, die digitale Transformations- und Innovationsfähigkeit von KMU in Baden-Württemberg nachhaltig zu steigern. Zu den Transferangeboten zählt auch die vorliegende Schriftenreihe, in welcher zu aktuellen Themenfeldern der Digitalisierung Bedarfe aufgedeckt, zukünftige Entwicklungsrichtungen untersucht und Lösungsansätze aufgezeigt werden.

Wir danken allen Experten, die an der Befragung teilgenommen haben und danken außerdem den Firmen Daimler und Bosch für den interessanten fachlichen Austausch. Weiterhin danken wir Sina Niedermaier für ihre Unterstützung und Zusammenarbeit.

Wir wünschen allen interessierten Leserinnen und Lesern eine spannende und inspirierende Lektüre und hoffen, dass Sie wichtige Erkenntnisse für die Wahl des passenden Werkzeugs zur Überwachung und Störungserkennung in vernetzten IT-Systemen gewinnen können.

Prof. Dr.-Ing. Prof. e. h. Wilhelm Bauer
Geschäftsführender Institutsleiter Fraunhofer IAO

Prof. Dr.-Ing. Oliver Riedel
Institutsleiter Fraunhofer IAO

Prof. Dr.-Ing. Anette Weisbecker
Stellvertretende Institutsleiterin Fraunhofer IAO

Jens Neuhüttler
Leiter des Business Innovation Engineering Center (BIEC)

Thomas Renner
Leiter des Forschungsbereichs Digital Business

2 Management Summary

Vernetzte IT-Systeme – Chance und Herausforderung. Die Vernetzung von IT-Systemen nimmt zu, nicht zuletzt durch mobile Endgeräte und intelligente Gegenstände (Internet of Things). In vernetzten IT-Systemen laufen zwischen einzelnen Akteuren verteilte Prozesse ab, die heutzutage für viele Dienstleistungen und zunehmend auch Produkte Kernfunktionen darstellen. Viele Systeme wandern in die Cloud, das heißt, Firmen betreiben keine eigene Hardware mehr, sondern mieten diese dynamisch nach Bedarf. Durch den Betrieb mehrerer Instanzen können Systeme hochskalierbar und hochverfügbar gemacht werden. Auf Anwenderseite bieten Dienste aus der Cloud den Vorteil, dass keine Installation oder Updates von Anwendungen mehr notwendig sind. Das macht es auch möglich, dass Dienste und Produkte beim Kunden reifen. Gleichzeitig bedeutet es aber auch, dass die Verfügbarkeit der verteilten Systeme gesichert sein muss, da der Anwender sonst seine Dienste und Produkte nicht nutzen kann.

Störungserkennung heute. Bei der überwachung dynamischer, cloud-basierter verteilter IT-Systeme stoßen konventionelle Überwachungswerkzeuge an ihre Grenzen. Da Störungen überall im verteilten System auftreten können, ist es schwierig, die Ursache einer Störung zu lokalisieren und diese zu beheben. Insbesondere in Bereichen wie IoT und Connected Car sind daher neue Lösungen zur Überwachung und Störungserkennung notwendig.

Methodik. Mittels Literaturstudie und Interviews von 29 Experten wurde der aktuelle state-of-the-art in der Störungserkennung verteilter IT-Systeme ermittelt. Basierend darauf wurden Herausforderungen, Chancen und Best Practices definiert.

Blick auf den Markt. Der Blick auf den Markt zeigt ein großes Angebot an reifen Softwarelösungen für Störungserkennung in verteilten IT-Systemen. Bestehende Technologien wie IT-Infrastuktur-Überwachung, Complex Event Processing und Log-Analyse haben ihren Funktionsumfang in den letzten Jahren erheblich erweitert, um mit neuen, komplexeren Cloud-Anwendungen umgehen zu können. Business Activity Monitoring und Business Process Monitoring haben als Technologien an Bedeutung verloren. Neue Technologien wie Application Performance Monitoring und Event Stream Analytics sind spezifisch auf die Bedarfe verteilter, cloud-basierter IT-Systeme ausgelegt.

Auswahl einer Softwarelösung. Es existiert keine Universallösung, die für alle Einsatzzwecke die optimale ist, sodass die Auswahl immer anhand individueller Rahmenbedingungen und Ziele erfolgen muss. Lösungen unterscheiden sich zum Teil erheblich in Featureumfang, Konfigurationsaufwand und Preis. Moderne Lösungen sind allerdings erweiterbar und kombinierbar. Die Eigenentwicklung von Monitoring-Software ist im Regelfall nicht mehr wirtschaftlich.

Zielgruppengerechte Überwachung. Bei der Umsetzung und dem Betrieb einer Störungserkennung in einem verteilten IT-System sind viele Stakeholder auf verschiedenen Unternehmensebenen involviert. Dazu gehören Unternehmensführung, Prozessverantwortliche, Entwicklung, Betrieb und Support. Außerhalb des Unternehmens kommen Kunden, Vertriebspartner und Dienstleister hinzu. Alle diese Gruppen haben verschiedene Anforderungen an eine Überwachungslösung und benötigen unterschiedliche Sichten und unterschiedliche Daten.

Fehlen von Standards. Es existieren Defizite bei Standards in Bezug auf Fehlermeldungen, Fehlercodes oder Log-Dateien. Abhilfe schaffen moderne Lösungen, indem sie eine Vielzahl von Datenformaten unterstützen. Im Regelfall kommen in verteilten Systemen verschiedene Datenformate zum Einsatz. Unternehmenseigene Standards können Vorteile bei der Störungserkennung und der Kommunikation von

Fehlern geben. Die Einführung solcher Standards ist aber auch in Hinsicht auf Systeme von Drittanbietern schwierig.

Ursachenerkennung. Für die automatische Erkennung der Ursache einer Störung existieren schon erste Ansätze, zum Beispiel für Netzwerkfehler. Für andere Fehler wie Logikfehler in Anwendungen ist eine Ursachenerkennung allerdings nur begrenzt möglich, und wenn dann nur mit hohem Aufwand.

Steigende Dynamik. Verteilte IT-Systeme zeichnen sich durch eine hohe Dynamik aus, da die beteiligten Komponenten und deren Software sich ständig ändern, z. B. durch Softwareupdates, Migration von Cloud-Servern oder neuen Nutzerendgeräten. Moderne Lösungen müssen diese Dynamik beherrschen, ohne, dass eine ständige manuelle Konfiguration notwendig ist.

Integration von Anwendungssystemen. Auf technischer Seite muss Software beobachtbar sein. Sie muss notwendige Kennwerte zur Überwachung nach außen sichtbar machen durch Schnittstellen, Logs, etc. Dazu sollte man moderne Architekturmuster und Datenformate verwenden.

Legacy-Systeme und Drittsysteme. Diese Systeme bieten nur begrenzte Daten zur Überwachung. Sind keine Schnittstellen vorhanden, müssen andere Wege gewählt oder die Software erweitert werden.

Standardlösung vs. Individuallösung. Es empfiehlt sich, grundsätzlich Standardsoftware einer Individuallösung vorzuziehen. Für komplexe und anwendungsspezifische Überwachung kann eine Entwicklung einer Individuallösung auf Basis von Standardtechnologien notwendig sein.

Störungserkennung als organisatorische Herausforderung. Störungserkennung ist eine unternehmensweite, abteilungsübergreifende Aufgabe. Oft scheitert diese nicht an technischen, sondern an organisatorischen Bedingungen. Gemeinsame Ziele und Strategien im Unternehmen müssen von der Geschäftsführung vorgegeben werden. Dafür ist eine Wertschätzung und ein Bewusstsein bei der Geschäftsführung notwendig. Störungserkennung muss schon während der Entwicklung berücksichtigt werden und alle Stakeholder sollten miteinbezogen werden. Bei der Definition von Zielen und Service-Leveln dürfen neben internen Anforderungen auch die Erwartungen der Kunden nicht vergessen werden. Im Betrieb sollten Kommunikationswege klar sein und einzelne Bereiche eng zusammenarbeiten, um Störungen schnell zu erkennen und zu beheben.

Störungserkennung als kontinuierlicher Prozess. Moderne Technologien als auch Methoden erlauben es, auch komplexe verteilte IT-Systeme zuverlässig zu betreiben und Störungen schnell zu beheben. Dies gelingt aber nicht auf Anhieb perfekt, egal, welche Softwarelösung gewählt wird. Überwachung und Störungserkennung sind eine Daueraufgabe und müssen kontinuierlich angepasst, verbessert und erweitert werden.

Sensibilisierung im Unternehmen. Eine schon oft vorhandene mangelnde Wertschätzung und ein mangelndes Bewusstsein im Unternehmen ist ein Hauptgrund für die unzureichende Etablierung und Integration einer Überwachung und Störungserkennung. Es fehlen oft klar definierte Strategien, Ziele und auch Verantwortlichkeiten. Dies ist Aufgabe der Führungsebene. Bei der Festlegung von Zielen und Service-Leveln ist es wichtig, die Kundenerwartungen in den Vordergrund zu stellen, denn deren Zufriedenheit und Entscheidung für ein Produkt oder Dienstleistung entscheidet den Erfolg, unabhängig davon, was vertraglich festgelegt ist. Überwachung und Störungserkennung sollten schon während der Entwicklung von Anwendungen berücksichtigt werden.

3 Einleitung

Die Vernetzung von IT-Systemen hat in den letzten Jahrzehnten unsere Welt entscheidend verändert. Inzwischen sind vernetzte IT-Systeme auch mobil, nicht nur in Form von Smartphones, sondern immer mehr auch in Form von intelligenten Gegenständen, beispielsweise Fahrzeugen. Vernetzte IT-Systeme bilden übergreifende Gesamtsysteme, das heißt, einzelne Systeme agieren als Teil eines größeren verteilten IT-Systems, ähnlich wie in einem Rechner einzelne Hardwarekomponenten zusammenarbeiten. In einem solchen verteilten IT-System werden Prozesse durchgeführt, in denen sich Anfragen, Antworten und Daten durch das System bewegen. So kommunizieren die einzelnen IT-Systeme im verteilten System, indem Nachrichten über verschiedene Protokolle (z. B. TCP, E-Mail) versendet werden (Kaplar et al., 2017). (Ghosh, 2014). Auch beschäftigt sich der Bereich Big Data viel mehr mit verteilten IT-Systemen (Freymann et al., 2020). Dies liegt auch daran, dass innerhalb eines verteilten IT-Systems eine viel höhere Datenkommunikation zwischen den Komponenten vorhanden ist.

Die Gründe für den Siegeszug von verteilten IT-Systemen sind vielfältig. Viele Anwendungen sind nur über ein verteiltes System möglich, weil sie Informationsaustausch und Kommunikation umfassen. Klassische Internetanwendungen wie Web-Browsing und E-Mail sind Beispiele dafür. Im Geschäftsbereich beschleunigt der elektronische Austausch von Daten, Dokumenten und Zahlungen die Prozesse und senkt so Kosten.

Des Weiteren wird das sogenannte Cloud-Computing immer stärker genutzt. Statt physische Hardware zu beschaffen, am Arbeitsort aufzustellen und zu betreiben, werden notwendige Rechnerkapazitäten über das Netzwerk gemietet. Ein wesentlicher Vorteil davon ist die beliebige Erweiterung von Rechenkapazität, die durch ein einfaches Hinzufügen von neuen CPU-Komponenten in das Gesamtsystem erreicht werden kann. Neben der Erweiterung der Rechenkapazität kann ein verteiltes IT-System auch in seinen Software- und Hardwareressourcen beliebig skaliert werden. Aber nicht nur eine Erweiterung der Rechen- oder Speicherkapazität ist möglich. Verteilte IT-Systeme ermöglichen beispielsweise auch ein Load Balancing und die Verteilung von Rechen- und Speicherkapazitäten (Freymann et al., 2020). Für geschäftskritische Anwendungen bietet ein verteiltes IT-System hohe Fehlertoleranz, die durch redundante Software- und Hardwarekomponenten erreicht wird. Fällt eine Soft- oder Hardware aus bzw. ist eine Störung aufgetreten, können andere Software- und Hardwarekomponenten den Fehler kompensieren, ohne dass das Gesamtsystem ausfällt und die Erreichbarkeit bzw. die Funktionalität beeinträchtigt wird. (Ghosh, 2014).

Auch bei der Bereitstellung von Software und Diensten für Endbenutzer bieten verteilte IT-Systeme einen Vorteil. So kann auf physische Installationsmedien verzichtet werden oder die Anwendung direkt zum Zeitpunkt der Benutzung geladen werden, zum Beispiel bei Browseranwendungen. Im Hintergrund können Prozesse und Algorithmen verbessert werden, ohne dass der Nutzer dazu Änderungen an seinem Gerät vornehmen muss. Dies ist insbesondere für vernetzte Produkte von Vorteil. So können z. B. für ein Fahrzeug neue Mehrwertdienste zur Verfügung gestellt werden, ohne dass das Fahrzeug dafür gewartet werden muss.

Mit dem Vorhandensein eines verteilten IT-Systems ergeben sich allerdings auch Herausforderungen, mit denen umgegangen werden muss. Dies liegt in der höheren Komplexität eines verteilten IT-Systems begründet, die sich durch eine höhere Anzahl von Komponenten, deren geographische (und zum Teil auch organisatorische) Verteilung sowie die Kommunikation zwischen den Komponenten ergibt. Besonders herausfordernd ist in diesem Zusammenhang die Dynamik als auch die Vernetzung der Komponenten in einem verteilten IT-System (Niedermaier et al., 2019).

Eine Hauptherausforderung ist das Behandeln von auftretenden Störungen, das aufgrund der höheren Komplexität eigene Methoden und Werkzeuge benötigt, weswegen die Störungsbehandlung in verteilten IT-Systemen ein eigenes Forschungsgebiet ist (Ghosh, 2014). Die vorliegende Studie beschäftigt sich mit der Behandlung von Störungen in verteilten IT-Systemen und gibt Anwendern und Entscheidern einen Überblick über aktuelle Lösungen zur Störungserkennung in verteilten IT-Systemen.

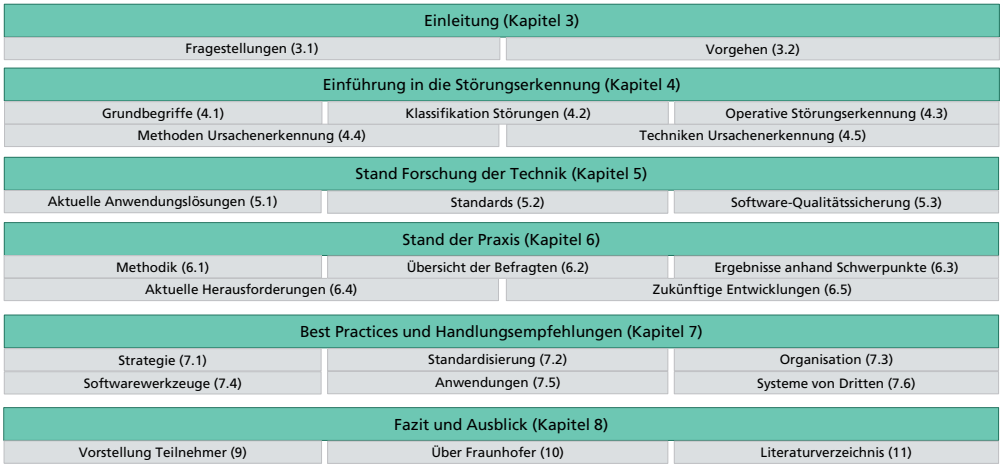

Abbildung 1: Aufbau der Studie

Der Aufbau der Studie ist wie in Abbildung 1 gezeigt: Nach einer Einleitung in Kapitel 3 in die Fragestellungen und das Vorgehen folgt in Kapitel 4 eine Einführung in die Störungserkennung. Zu der Einführung gehören unter anderem Grundbegriffe, die Klassifikation der Störungen oder auch die Methoden der Ursachenerkennung. Kapitel 5 beschäftigt sich dann mit dem Stand der Forschung in den Bereichen Standards, Softwarequalitätssicherung und den aktuellen Anwendungslösungen. Nach der Beschreibung des Forschungsstandes wird in Kapitel 6 der Stand der Praxis anhand der Ergebnisse der Experteninterviews beschrieben. In Kapitel 7 werden dann aus den Interviews abgeleitete Best Practices und Handlungsempfehlungen beschrieben. In Summe werden sechs Bereiche betrachtet: Strategien, Standardisierungen, Organisation, Softwarewerkzeuge, Anwendungen und Systeme von Dritten. Zum Schluss gibt Kapitel 8 ein Fazit und einen Ausblick.

3.1 Fragestellungen

Die vorliegende Studie betrachtet die Behandlung von Störungen in verteilten IT-Systemen. Dies soll Anwendern und Entscheidern einen Überblick über aktuelle Lösungen zur Störungserkennung geben. Des Weiteren geht es um Darlegung der Erfahrungen in der Praxis bzw. Industrie zur Überwachung und Störungserkennung. Mit folgenden Fragestellungen beschäftigt sich die Studie:

Frage	Inhalt
Frage 1	*Wie ist der Stand der Forschung und Technik bei der Störungserkennung?*
Frage 2	*Wie setzt die Praxis die Störungserkennung in verteilten IT-Systemen um?*
Frage 3	*Was sind aktuell die Best Practices, um eine Störungserkennung in verteilten IT-Systemen zu etablieren?*
Frage 4	*Was sind aktuell die Handlungsempfehlungen, um eine Störungserkennung in verteilten IT-Systemen zu etablieren?*

Tabelle 1: Fragestellungen der Studie

3.2 Vorgehen

Um diese Fragestellungen zu beantworten, wurden zwei Analysen durchgeführt. Zu Beginn steht eine state-of-the-art-Analyse in Form einer Literaturstudie, die sich mit den Grundlagen der Störungserkennung und mit dem Stand der Forschung und Technik beschäftigt (Kapitel 4 und 5). Betrachtet wurden wissenschaftliche Literatur und Studien[1], sowie Anbieterinformationen mit der Berücksichtigung existierender und neuer Player am Markt. Darauf aufbauend wird im zweiten Schritt der Stand der Praxis untersucht, um zu ermitteln, welche Herausforderungen in der Praxis bestehen und was gängige Methoden und Techniken sind, um Störungen in verteilten IT-Systemen zu erkennen (Kapitel 6). Dazu wurden semi-strukturierte Interviews mit Experten auf Anwender- und Anbieterseite durchgeführt.

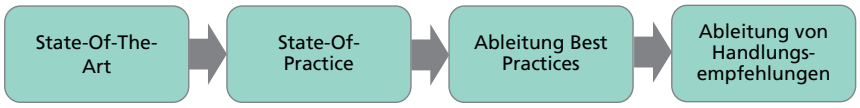

Abbildung 2: Übersicht über das Vorgehen der Studie

Mit den gewonnenen Erkenntnissen werden darauffolgend Best Practices und Handlungsempfehlungen für Störungserkennung in verteilten IT-Systemen abgeleitet.

[1] Vidačković et al. (2010) und Kötter et al. (2014).

4 Einführung in die Störungserkennung

Der zuverlässige Betrieb von Softwaresystemen hat die Informatik schon von früh an beschäftigt. So stammt beispielsweise der Begriff »Bug« der Informatik-Folklore zufolge wortwörtlich von einer Motte, die sich in einem Relais des Mark II Großrechners verfing und zu einem Rechenfehler führte. Hier zeigt sich schon, dass ein Software-Bug nicht nur in der Softwareentwicklung entsteht, sondern auch im Betrieb auftreten kann. Daher reicht es nicht aus, eine Software während des Entwicklungsprozesses zu testen, sondern es muss auch eine Qualitätssicherung des Softwaresystems im Betrieb stattfinden. Dafür ist eine Überwachung und Störungserkennung unerlässlich. Dieses Kapitel enthält eine kurze Einführung in die wichtigsten Begriffe und Themen dieser Studie.

4.1 Grundbegriffe

Bevor über Fehler und Störungen gesprochen werden kann, ist es zunächst wichtig, diese Begriffe klar zu definieren und voneinander abzugrenzen. In der (meist englischen) Fachliteratur werden drei verschiedene Begriffe unterschieden, wie Tabelle 2 zeigt:

Grundbegriff	Beschreibung
Irrtum *(error)*	*Ein in der Software vorhandener Fehler. Beispiel: Software wurde nicht für negative Rechnungsbeträge (Rückerstattungen) ausgelegt.*
Fehlzustand *(fault / defect)*	*Ein fehlerhafter Zustand im Softwarebetrieb. Beispiel: Falls ein negativer Rechnungsbetrag in einer Bestellung vorhanden ist, wird keine Rechnung für die Bestellung angelegt.*
Ausfall *(failure)*	*Ein nach außen sichtbarer Fehler der Software. Beispiel: Sobald der Kunde eine Bestellung mit Gutschrift durchführen möchte, erhält er eine Fehlermeldung, dass keine Rechnung gefunden wurde.*

Tabelle 2: Grundbegriffe für Fehler und Störungen

Ein Irrtum kann also einen Fehlzustand bewirken, aus dem ein Ausfall folgt. Im Deutschen ist eine Abgrenzung zwischen »error«, »fault« und »failure« leider nicht so klar möglich, da sich alles drei als »Fehler« übersetzen lässt. Daher wird im Rahmen dieser Studie eine ähnliche, aber im Deutschen klarere Definition verwendet, wie Tabelle 3 und Abbildung 3 zeigen.

Grundbegriff	Beschreibung
Fehlerursache *(error cause)*	*Die zugrundeliegende Ursache für einen Fehlerzustand, z. B. Stromausfall, Software-Bug, fehlende Netzwerkverbindung, fehlerhafte Konfiguration, Überlastung. Manchmal auch: Fehlbedienung/Fehlannahme des Nutzers.*
Fehler *(error)*	*Ein Fehlerzustand in einem System, das heißt, eine Abweichung des Ist-Zustands vom Soll-Zustand, z. B. System ist offline, zu hohe Antwortzeiten, fehlerhafte Antworten, ausbleibende Antworten.*
Störung *(malfunction)*	*Eine von außen sichtbare Fehlerwirkung des Systems, die dessen Funktion beeinträchtigt. Z. B. Login nicht möglich, keine grafische Anzeige, unzumutbare Antwortzeiten, GUI reagiert nicht, Anzeige falscher Daten.*

Tabelle 3: Grundbegriffe für Ursache und Auswirkung eines Fehlers

Nach dieser Definition wäre die Störungserkennung die Erkennung nach außen sichtbarer Fehlerwirkungen und Funktionsbeeinträchtigungen von Softwaresystemen. Dies greift jedoch zu kurz. Wenn lediglich sichtbare Fehlerwirkungen erkannt werden, ist dies nur ein beschränkter Erkenntnisgewinn über eine rein menschliche Beobachtung des Systems hinaus. Falls solche auftreten, werden die Nutzer des Systems diese vermutlich melden, auch wenn eine Software dies natürlich schneller kann. Die Störungserkennung geht über die reine Erkennung von Störungen nach dieser Definition hinaus.

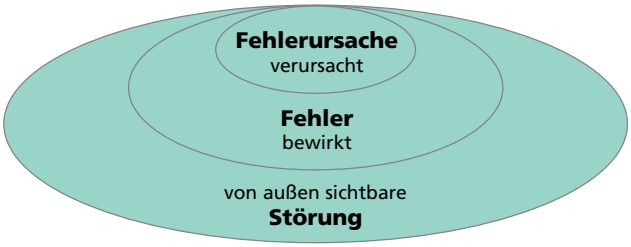

Abbildung 3: Fehlerursache, Fehler und Störung[2]

Neben Störungen sollen auch Fehler erkannt werden, möglichst, sobald diese auftreten und noch bevor sie eine Störung bewirken. Im Idealfall wäre eine Störungserkennung sogar in der Lage, zu einem auftretenden Fehler dessen Fehlerursache zu nennen (sogenannte Ursachenerkennung oder engl. Root Cause Detection). Störungserkennung wird infolgedessen wie folgt definiert:

Als Störungserkennung werden Werkzeuge und Maßnahmen bezeichnet, die dazu dienen, Fehlerursachen, Fehler und Störungen zu erkennen, die im Betrieb von Softwaresystemen auftreten.

4.2 Klassifikation von Störungen in verteilten IT-Systemen

Beim Entwurf und im Betrieb von verteilten Softwaresystemen steht generell die Frage im Raum, welche Arten von Störungen überhaupt auftreten können. Dies ist wichtig dafür, zu entscheiden, welche Störungen vermieden werden, wie Störungen erkannt und letztendlich wie sie behoben werden können. In der Fachliteratur gibt es verschiedene Arten, Störungen und Softwarefehler zu klassifizieren. Einen sehr umfangreichen Katalog von Softwarefehlern stellt beispielsweise die BITKOM zur Verfügung (BITKOM, 2007). Im Folgenden werden zwei allgemeine Klassifikationen vorgestellt, die geeignet sind mögliche Störungen in einem konkreten verteilten IT-System zu identifizieren. Die erste behandelt die Komponenten eines verteilten IT-Systems, die zweite die Kommunikationskanäle. Der Begriff Fehlersemantik bezeichnet das Verhalten von Systemkomponenten im Fehlerfall. Folgende Klassen von Störungsverhalten (Tabelle 4) sind dabei möglich (siehe auch Abbildung 4):

Fehlerklasse	Beschreibung
Byzantinischer Fehler *(byzantine failure)*	*Das System liefert beliebige Antworten (richtige oder falsche oder gar keine, auch mehrere auf eine Anfrage, auch Antworten ohne Anfrage). Meistens komplexere Fehler in innerer Logik des Systems.* *Beispiel: Die Benutzeroberfläche zeigt »Kauderwelsch« an, es werden Nachrichten an den falschen Empfänger gesendet.*

[2] angelehnt an (Spillner et al., 2008)

Falschantwort *(response failure)*	*Eine falsche Antwort auf eine Anfrage wird geliefert.* *Beispiel: Eine nicht vorrätige Ware wird als vorrätig angezeigt, der Gesamtbetrag einer Rechnung ist nicht korrekt.*
Performance-fehler *(performance failure)*	*Eine Antwort wird zu spät versendet (in Bezug auf Service Level Agreements, Time-Outs, etc.).* *Beispiel: Der Aufruf des E-Mail-Postfachs dauert mehrere Minuten, das Navigationssystem sagt die Ausfahrt erst an, nachdem sie bereits passiert worden ist.*
Dienst-verweigerung *(omission failure)*	*Eine Nachricht wird nicht versandt / eine Anfrage wird nicht beantwortet.* *Beispiel: Ein Dokument wird nicht ausgedruckt, eine E-Mail kommt nicht am Ziel an, eine Bestellung wird nicht entgegengenommen.*
Absturz *(crash failure)*	*Eine Anfrage wird nicht beantwortet. Die Systemkomponente antwortet nicht mehr auf zukünftige Anfragen.* *Beispiel: Das Browserfenster friert ein, der Server reagiert nicht mehr auf Anfragen, der Server stoppt mit einem »Blue Screen of Death«.*
Stop *(fail-stop failure)*	*Eine Anfrage wird nicht beantwortet. Die Systemkomponente antwortet nicht mehr auf zukünftige Anfragen. Die Systemkomponente ist ausgeschaltet.* *Beispiel: Das Browserfenster friert ein, der Server reagiert nicht mehr auf Anfragen. Wie ein Absturz, allerdings wird das System geordnet gestoppt. Dadurch ist erkennbar, dass das System nicht mehr läuft.* *Beispiel: Das Browserfenster schließt sich abrupt, der Rechner schaltet sich ab.*

Tabelle 4: Übersicht der Klassen von Störungsverhalten

Diese Störungsklassen bilden eine Hierarchie. Jeder Fehlerfall ist ein Spezialfall des Vorhergehenden. So ist zum Beispiel der Absturz eines Systems ebenfalls eine Dienstverweigerung (keine Antwort), ein Performancefehler (mit unendlicher Wartezeit) Falschantwort (richtige Antwort wird nicht geliefert) und ein byzantinischer Fehler.

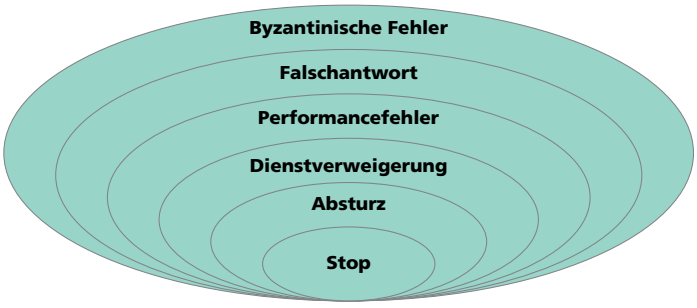

Abbildung 4: Hierarchie von einzelnen Störungsklassen[3]

[3] angelehnt an (Spillner et al. 2008)

Umgekehrt gilt, dass je allgemeiner eine Fehlerklasse ist, desto schwerer ist sie zu erkennen. Ein Stop-Fehler eines Webservers lässt sich beispielsweise durch einfache technische Maßnahmen wie einen Ping (Verfügbarkeitsanfrage über das Netzwerk) erkennen. Für einen Performance-Fehler sind schon komplexere Methoden notwendig wie z. B. Messen der Antwortzeiten und Überwachen des Durchschnittswerts über die letzten zehn Minuten. Ein byzantinischer Fehler hingegen ist nur sehr schwer automatisch zu erkennen, da er die gesamte spezifizierte Funktion der Software betreffen kann. Hier können Methoden wie Plausibilitätsprüfungen eine Erkennung bestimmter byzantinischer Fehler ermöglichen. Eine andere Art, das Störungsverhalten von Systemen zu klassifizieren, ist es, nach Art der Störungserkennung (Tabelle 5) zu unterscheiden (Arshad, 2014):

Störungsverhalten (System)	Beschreibung
Still *(Silent)*	*Stille Fehler, die keine Fehlerausgabe haben. Keine Logs von Fehlern sind in der Systemkomponente.* *Beispiel: Ein falscher Wert wird in eine Datenbanktabelle geschrieben, eine Datei wird ungefragt im Hintergrund gelöscht.*
Nicht-Still *(Non-Silent)*	*Keine Fehlerausgabe, aber evtl. offensichtliches Fehlverhalten. Logs von Fehlern sind in der Systemkomponente.* *Beispiel: Bluescreen of Death, System friert ein und reagiert nicht mehr auf Eingaben, Anwendung schließt sich abrupt.*
Nicht-Still interaktiv *(Non-Silent Interactive)*	*Nutzer erhält eine aussagekräftige Fehlermeldung. Logs von Fehlern sind in der Systemkomponente.* *Beispiel: System teilt mit, dass der Hauptspeicher nicht ausreicht, System teilt mit, dass ein Vorgang nicht abgeschlossen werden kann.*

Tabelle 5: Unterscheidung nach Art der Störungserkennung

Während nicht-stille Störungen für den Nutzer des Systems oder den Administrator (durch Log-Dateien) offenkundig sind, können stille Störungen nur schwerer erkannt werden, ähnlich den byzantinischen Fehlern. Daraus ergibt sich, dass nicht-stille Fehler mit einfachen technischen Maßnahmen zeitnah erkannt werden können, während für stille Fehler komplexere Lösungen notwendig sind.

Neben dem Störungsverhalten von Systemen lässt sich auch das Störungsverhalten von Nachrichtenkanälen (wie z. B. Netzwerkverbindungen) als Fehlersemantik beschreiben (Tabelle 6) (Mattern, 2005):

Störungsverhalten (Nachrichtenkanäle)	Beschreibung
Vielleicht *(Maybe)*	*Es wird einmal versucht, eine Nachricht zu schicken. Einfaches Verfahren, aber keine Erfolgsgarantie.*
Mindestens einmal *(At-least-once)*	*Versand wird wiederholt, bis er erfolgreich war. Möglicherweise wird Nachricht auch mehrfach gesendet. In der Praxis oft mit einem Timeout (d.h. der Versand erfolgt nicht mehr, wenn er bis zu einem bestimmten Zeitpunkt nicht erfolgt ist).*

Höchstens einmal *(At-most-once)*	*Versand wird wiederholt, bis er erfolgreich war. Auch werden Duplikate beim Empfänger erkannt. In der Praxis oft mit einem Time-Out.*
Exakt einmal *(Exactly-once)*	*Nachricht wird genau einmal erfolgreich versandt. In der Praxis nicht möglich, da dafür eine perfekte Verbindung notwendig wäre.*

Tabelle 6: Übersicht Störungsverhalten von Nachrichtenkanälen

Während es auf technischer Ebene in einem Nachrichtenkanal auch zu Veränderungen einer Nachricht kommen kann (bei Übertragungsfehlern), so ist dies bei üblichen digitalen Übertragungswegen heutzutage durch Prüfsummen und ähnliche Verfahren quasi ausgeschlossen. Dabei ist jedoch zu beachten, dass dies nicht für zwischengelagerte Systeme auf dem Übertragungsweg, wie z. B. Routern gilt. Bei diesen Systemen können alle zuvor beschriebenen Fehler bis hin zu byzantinischen Fehlern vorkommen (z. B. im Fall, dass die Komponente durch Hacking o.Ä. kompromittiert wurde). Was bedeuten nun diese Störungssemantiken für die Praxis?

Schon im Entwurf eines verteilten IT-Systems können die Störungssemantiken dazu verwendet werden, mögliche Störungen zu erkennen. Für jedes System und jeden Nachrichtenkanal sollte eine Klassifizierung möglicher Fehler anhand der Fehlersemantik erfolgen:

- Welche Arten von Störungen sind möglich?
- Welche Auswirkungen haben diese Störungen auf das gesamte System?
- Wie kann ich diese Störungen erkennen?
- Wie kann ich diese Störungen behandeln (automatisch oder manuell)?

Darauf basierend lassen sich technische und organisatorische Maßnahmen festlegen. Diese Maßnahmen betreffen:

- Störungserkennung (z. B. durch Monitoring-Software, Plausibilitätsprüfung)
- Störungsbehebung (z. B. durch manuellen Entstörungsprozess)
- Störungskompensation (z. B. durch redundante Systeme und Netzwerke)

Dabei ist das Gelingen unwahrscheinlich, alle möglichen Störungsarten durch automatische Lösungen abzudecken (insbesondere für byzantinische Fehler). Hier ist zu überlegen, wie viel manueller Aufwand zur Störungserkennung im Betrieb angemessen ist. In Abbildung 5 wird dieser Zusammenhang veranschaulicht. Daraus kann dann auch abgeleitet werden, welches akzeptable Restrisiko verbleibt. Dabei gibt es kein Richtig oder Falsch, denn es hängt stark vom Geschäftszweck des verteilten IT-Systems ab. So ist in der Finanzbranche beispielsweise ein höherer Aufwand für den sicheren Betrieb notwendig und ein wesentlich geringeres Restrisiko akzeptabel als in anderen Branchen.

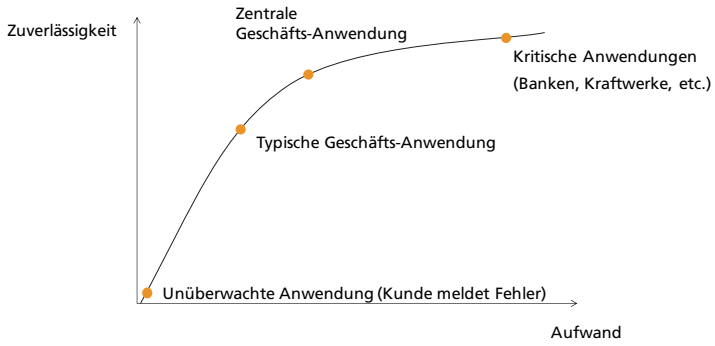

Abbildung 5: Verhältnis von Aufwand zu Nutzen bei der Störungserkennung

4.3 Operative Störungserkennung und Monitoring

Neben einer technischen Funktion ist Störungserkennung auch eine Tätigkeit im Unternehmen. Hierbei wird auch von operativer Überwachung bzw. operativem Monitoring gesprochen. Grundsätzlich sind dabei zwei Ansätze zu unterscheiden:

Monitoringart	Beschreibung
Aktives Monitoring	*Das Personal beobachtet ein System, um Fehler und Störungen zu entdecken, sobald diese auftreten, z. B. über Dashboards.*
Passives Monitoring	*Ein System benachrichtigt das Personal über potenzielle Störungen, z.B. über E-Mail-Alarme.*

Tabelle 7: Arten von operativem Monitoring

In Tabelle 7: Arten von operativem Monitoring sind die beiden grundlegenden Arten von operativem Monitoring aufgeführt. Das aktive Monitoring bindet Personalressourcen und ist dadurch mit hohem Kostenaufwand verbunden, dies kann für kritische Infrastrukturen allerdings notwendig sein. Auch ist es technisch weniger anspruchsvoll als passives Monitoring, da der Mensch hier Aufgaben übernimmt, die sonst automatisiert werden müssten.

Mit der wachsenden Komplexität verteilter IT-Systeme und steigenden Ansprüchen an Verfügbarkeit und Service-Level spielt das passive Monitoring eine immer größere Rolle.

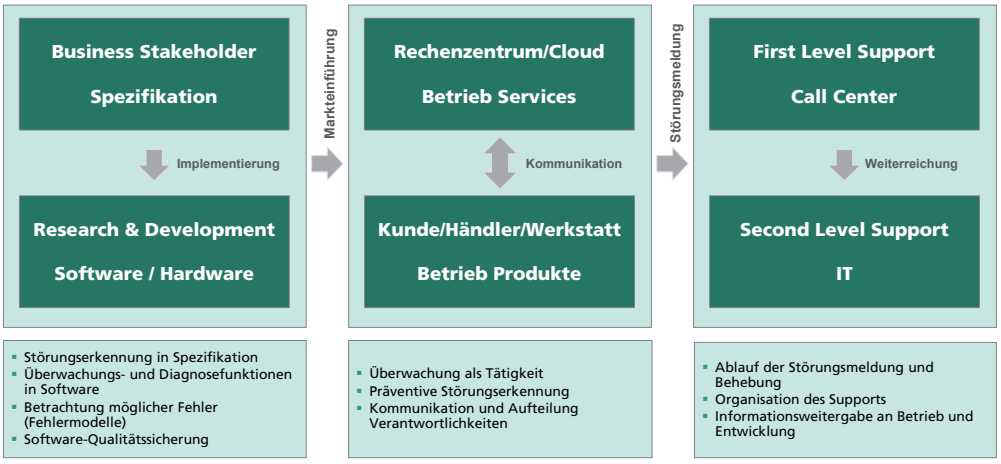

Abbildung 6: Beispielhafte Organisationsstruktur für operatives Monitoring

Abbildung 6 zeigt eine beispielhafte Organisationsstruktur für die operative Überwachung eines Produkts. In die Störungserkennung und Behebung sind verschiedene Teile des Unternehmens involviert. Die Herausforderung ist, trotzdem eine gemeinsame Störungserkennung und Behebung durchzuführen. Dies schließt insbesondere ein, dass Verantwortlichkeiten, Verantwortliche und Prozesse klar definiert und kommuniziert sind. Damit verbunden ist die Definition klarer Kommunikationswege mit dem Kunden, sodass dieser einen Ansprechpartner hat, von dem er Hilfe erhält und nicht durch das Unternehmen »durchgereicht« wird (One-Face-To-The-Customer).

In verteilten IT-Systemen sind oft verschiedene Unternehmensbereiche, z.T. sogar verschiedene Unternehmen, involviert. Dadurch gibt es nicht nur externe Kunden, sondern auch firmeninterne Kunden, die Teile ihres Softwareprodukts bzw. ihrer Dienstleistung von einem anderen Teil des Unternehmens beziehen. Genau wie bei einem externen Kunden müssen hier klare Kommunikationswege und auch Service-Level definiert sein.

Verschiedene Stakeholder haben verschiedene Anforderungen an das operative Monitoring, die darin begründet sind, dass sie verschiedene Verantwortungen und damit auch verschiedene Definitionen einer Störung haben.

Für die Geschäftsleitung ist es wichtig, aktuelle Geschäftskennzahlen zu kennen, um sicherzustellen, dass Prozesse im wirtschaftlichen Sinne ordnungsgemäß verlaufen. Der Prozessverantwortliche muss hingegen wissen, dass die Prozesse korrekt ablaufen und benötigt dafür Kennwerte wie Produktionszahlen, Lagerbestände und Auftragsvolumen. Die IT-Abteilung ist dagegen an technischen Kennzahlen interessiert, um sicherzustellen, dass Software- und Hardwaresysteme ordnungsgemäß funktionieren. Das umfasst Kennwerte wie Systemverfügbarkeit, Antwortzeiten und Systemauslastung. Der Kundenservice benötigt eine andere Sicht, denn für den Kontakt mit dem Kunden steht die Behebung im Vordergrund. Hier ist es wichtig, zu wissen, warum eine Funktion für den Kunden nicht funktioniert und wie Abhilfe geschaffen werden kann.

Ist eine Störung behoben, sollte ein Anschlussprozess zur Optimierung stattfinden. Dabei prüft man, ob diese Störung in Zukunft besser behandelt werden kann. Dies kann auf mehreren Wegen geschehen, zum Beispiel durch Erweiterung der Dokumentation, einen zusätzlichen Kennwert im Monitoring-System oder eine neue Version der Software.

4.4 Methoden der Ursachenerkennung

Neben der eigentlichen Erkennung von Störungen ist die Suche nach der eigentlichen Ursache der Störung wesentlich für die Störungsbehebung. Ist die Ursache bekannt, können weitere Störungen schneller erkannt oder aber auch vermieden werden. Jedoch zeigt die Praxis, dass Ursachenerkennung keine leichte Aufgabe ist. Alleine die Fehlersuche kann schwierig sein (Mace et al., 2016), denn aufgrund der Komplexität der Systeme sind Abläufe nur schwer nachzuvollziehen (Zvara et al., 2017). Somit erfordert die Ursachenerkennung richtige Werkzeuge, Wissen und Technologien, die korrekt eingesetzt werden müssen.

Es gibt verschiedene Herangehensweisen und Techniken, um Ursachen von Störungen in verteilten Systemen zu erkennen (Zawawy et al., 2010). Abbildung 7 zeigt vier verschiedene Methoden, die in der Praxis erfahrungsgemäß weite Verbreitung finden. Dazu gehören das Logging, Tracing, Detecting und das Monitoring, wobei Letzteres das Logging, Tracing und Detecting oftmals nutzt.

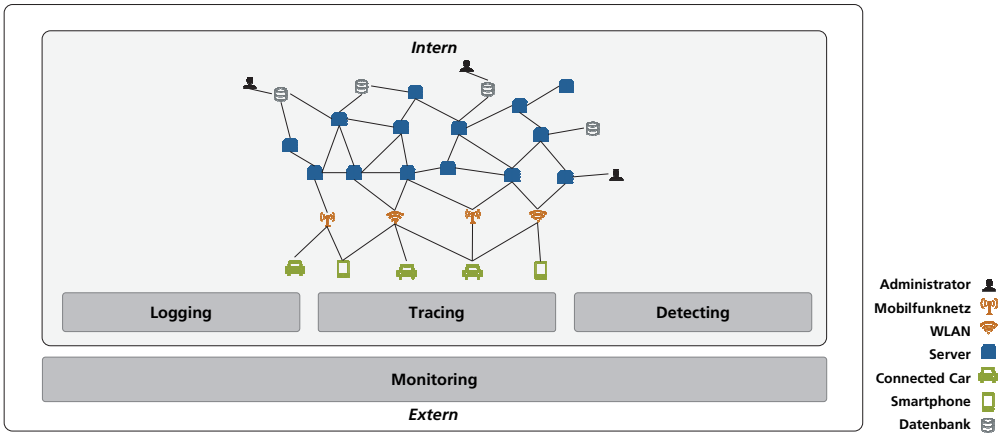

Abbildung 7: Methoden für die Ursachenerkennung von Störungen

4.4.1 Logging-Methode

Bei der Logging-Methode erzeugt jede Komponente im verteilten IT-System (z. B. Server, Router, Edge-Gerät oder die Datenbank) Log-Einträge. Das heißt, sie protokolliert Geschehnisse wie beispielsweise eingehende Anfragen, Antworten, Entscheidungen und Fehler. Im einfachsten Fall werden diese Einträge in Log-Dateien geschrieben, die auf der jeweiligen Komponente gespeichert sind. Dies erschwert allerdings den Zugriff auf Log-Einträge sowie die Kombination von Log-Einträgen verschiedener Komponenten, sodass eine Gesamtsicht auf das System nur mit erheblichem manuellem Aufwand möglich ist. Hier zeigt die Praxis, dass ein zentraler Speicherort sinnvoll ist, beispielsweise mittels einer Log-Datenbank.

Aber auch mit einem zentralen Log ist es schwierig, dieses auch zu sichten, denn die Anzahl der Einträge kann in größeren Systemen in die Millionen gehen. Ein klassischer Ansatz, um dieser Komplexität Herr zu werden, sind Log-File-Viewer, die ein strukturiertes Betrachten und Filtern der Dateien ermöglichen. Darüber hinaus ist allerdings eine automatische Analyse von Log-Dateien sinnvoll, da diese ein passives Monitoring ermöglicht.

Die am meisten verwendete Herangehensweise ist die Analyse der Log-Dateien so zu gestalten, dass ein Unterschied (Delta) zwischen einem normalen und nicht normalen Verhalten (Muster) erkannt wird, mit dem darauffolgend Ursachen identifiziert werden können (Zawawy et al., 2010). Hier kommen Machine-Learning-Techniken aber auch statistische Analysen zum Einsatz (Mace et al., 2016). Geloggt werden können beispielsweise erfolgreiche bzw. nicht erfolgreiche Transaktionen, Ein- und Ausgänge von Datenpaketen oder auch Ergebnisse von Einzelschritten zur Verarbeitung von Anfragen. Wenn Drittanbieter mit in einem verteilten IT-System integriert sind, dann ist es notwendig, diese Log-Daten über eine Schnittstelle anzubieten.

Ablauf der Logging-Methode. In Abbildung 8 wird dieser Zusammenhang veranschaulicht. Hier sendet beispielsweise ein Auto ein Signal, das über mehrere Schritte (grüner Pfad) über Server, Datenbanken und Schnittstellen transportiert wird. Log-Einträge, die eine Störung festhalten, sind rot umrahmt. Alle Log-Daten werden in einer zentralen Datenbank gespeichert. Die in der Log-Datenbank gespeicherten Log-Einträge werden dann mit unterschiedlichen Methoden analysiert, um Anomalien, Störungen, und deren Ursachen zu erkennen und auch, um Voraussagen treffen zu können. Das gängige Verfahren ist, Abweichungen vom normalen erwarteten Verhalten der Komponenten zu erkennen. Als Beispiel ist in der Abbildung 8 als Problem eine nicht durchführbare Datenspeicherung identifiziert worden. Darauffolgend ist es dann notwendig, die Ursache für diese Störung herauszufinden. Aufgrund der Log-Einträge lässt sich beispielsweise der Ausfall eines Servers identifizieren.

Vorteile der Logging-Methode. Die Logging-Methode ist grundsätzlich eine gängige Methode für die Überwachung und Störungserkennung von verteilten IT-Systemen. Auch wird diese Methode in der Praxis gerne benutzt, um Ursachen für Performance-Probleme zu finden. Jede übliche Komponente erzeugt Log-Dateien. Diese stellen insofern einen »kleinsten gemeinsamen Nenner« der Datenquellen dar. Es existieren zahlreiche Open-Source Standards und Frameworks.

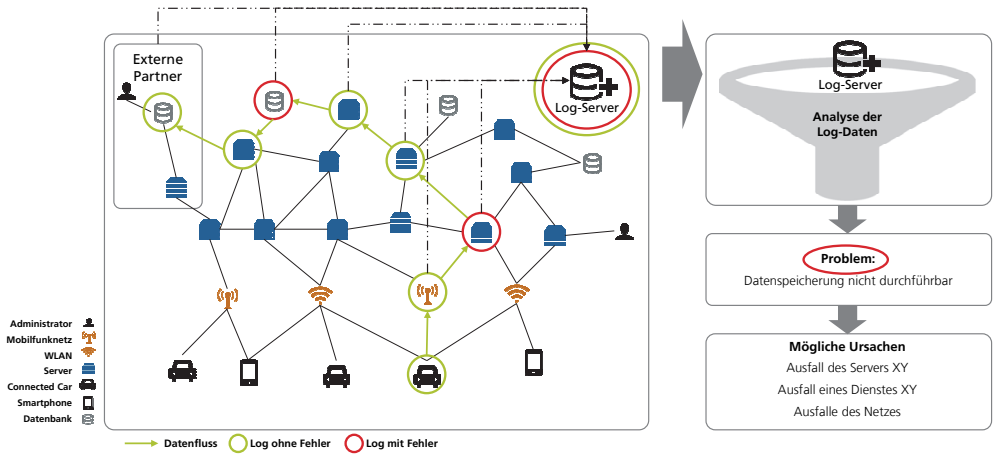

Abbildung 8: Übersicht über die Logging-Methode

Herausforderungen der Logging-Methode. Auch wenn die Logging-Methode häufig eingesetzt wird, sind jedoch auch einige Herausforderung mit dieser Methode verbunden.

Eine Herausforderung des Loggings ist die Fülle der anfallenden und zu analysierenden Daten. Dies kann eine große Komplexität erzeugen, die die Durchführung von Analysen selbst mit Expertenwissen schwierig macht. Auch ist die Auswahl von geeigneten Tools und Techniken für die Analysen der Log-Daten eine Herausforderung. Um die Komplexität des Diagnoseprozesses zu verringern ist es notwendig, die Log-Daten zu reduzieren und zu aggregieren. (Zawawy et al., 2010)

4.4.2 Tracing-Methode

Die Tracing-Methode hat zum Ziel, eine Anfrage auf ihrem Weg durch das verteilte IT-System von Beginn bis zum Ende zu verfolgen. Tracing hilft so, Abläufe und Verbindungen in verteilten IT-Systemen besser zu verstehen (Anderson et al., 2009). Zusammenhängende Informationen einer Nachricht werden auf ihrem Weg gesammelt (Mace et al., 2016). Dies kann beispielsweise sein, ob eine Datenweiterleitung erfolgreich oder nicht erfolgreich stattgefunden hat. Gesammelt werden diese Daten bzw. Informationen an bestimmten, im System vorhandenen Komponenten (Knotenpunkten), wie z. B. Servern oder Routern (Mace et al., 2016).

Ablauf der Tracing-Methode. In Abbildung 9 wird das Tracing einer Anfrage abgebildet. Ein Auto möchte Daten in einer Datenbank speichern und sendet die entsprechende Anfrage in das verteilte IT-System. An jedem Knotenpunkt wird in einem eigenen Datensatz festgehalten, was an dem entsprechenden Knotenpunkt passiert ist. Wie zu sehen ist, geschieht zwischen dem Knotenpunkt 4 und 5 etwas, weswegen Daten nicht weiterverarbeitet werden können. Aus diesem Grund kann am Kontenpunkt 10 der entsprechende Datensatz nicht gespeichert werden.

So wird generell der Ablauf durch das System hinweg aufgezeichnet: der Weg einer Anfrage, Antwort oder Transaktion. Es handelt sich um eine zur Laufzeit gesammelte Information, die von jedem verschiedenen Knotenpunkt Kausalzusammenhänge zwischen der Vergangenheit, Gegenwart und Zukunft der Nachricht beinhaltet (Zvara et al., 2017). Die Tracing-Daten werden gewöhnlich mittels eines Services separat gesammelt und ausgewertet (Zvara et al., 2017). Mittels der Tracing-Daten kann nach der Ursache einer Störung geforscht werden. Oft wird Tracing mit anderen Methoden verknüpft, wie Monitoring und Logging.

Vorteile der Tracing-Methode. Die Tracing-Methode ist ein gängiger Ansatz, um verteilte IT-Systeme zu überwachen. Auf dem Markt finden sich viele Lösungen (Zvara et al., 2017). Die Methode wird auch benutzt, um Ursachen für Performance-Probleme zu finden. Grundsätzlich werden zwei Richtungen des Tracings unterschieden: Vorwärts- und Rückwärts-Tracing (Zvara et al., 2017). Hier werden entweder Verbindungen zum Vorgänger oder Nachfolger gezogen.

Herausforderungen der Tracing-Methode. Die Tracing-Methode erfordert einen gewissen Aufwand, diese in einem verteilten IT-System mit unterschiedlichen Technologien und Programmiersprachen zu implementieren. Eine Herausforderung ist die Auswertung von den Tracing-Daten, wenn keine geeigneten Werkzeuge zur Verfügung stehen, um die Fülle der Daten zu analysieren. Des Weiteren muss die Implementierung des Tracings effizient gestaltet sein, da sonst die Performance des Systems beeinträchtigt sein kann. Somit sollte auch das gesamte verteilte IT-System für Tracing ausgelegt sein, um eine Überwachung mittels Tracings zu ermöglichen. Des Weiteren kann es passieren, dass nur die lokale Zeit der Knotenpunkte und nicht die globale Zeit des Gesamtsystems erfasst wird, was zu Zeitdifferenzen eines Traces führen kann, die die Auswertung durch scheinbar widersprüchliche Zeitangaben erschweren (Anderson et al., 2009).

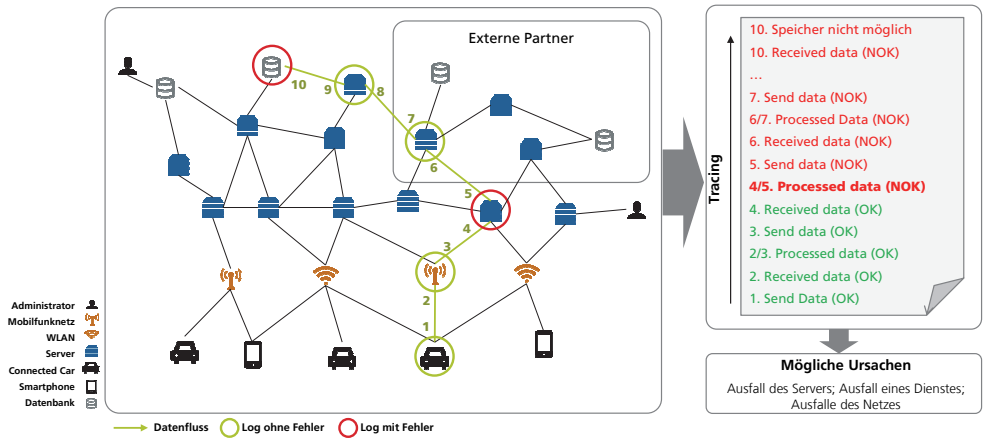

Abbildung 9: Übersicht über die Tracing-Methode

4.4.3 Detecting-Methode

Bei dieser Methode werden keine Daten aufgezeichnet wie beim Logging oder Tracing, sondern es werden an den entsprechenden Knotenpunkten Detektoren (Agenten) eingesetzt. Mittels dieser Agenten können beispielsweise Daten über den Netzwerkfluss erhoben werden (Du et al., 2003), die Fehler direkt erkennen, wie z. B. den Ausfall von Servern oder Diensten. Detektoren können Software-Komponenten sein (Kufel, 2016), die z. B. den Datendurchsatz oder die Verbindungsqualität messen oder Ausfälle zeitnah erkennen. Detektoren als Hardware-Komponenten sind Sensoren, die beispielsweise die Temperatur oder die CPU-Auslastung messen. Wird bei den Software- oder Hardwarekomponenten eine Anomalie detektiert, geben diese sofortiges Feedback.

Vorteile der Detecting-Methode. Im Vergleich zu der Logging- oder Tracing-Methode müssen vorab keine Log- oder Tracing-Daten ausgewertet werden, sondern Störungen werden direkt erkannt, weil die Agenten Einheiten für die Fehlererkennung darstellen (Gärtner, 1999). Des Weiteren können durch den Agenten viele interne Metriken gesammelt werden, die einen tiefen Einblick in die Komponente und somit auch in das Gesamtsystem erlauben (Kufel, 2016). Dies vereinfacht die Ursachenanalyse. So kann mit dieser Methode proaktiv reagiert werden und es vergeht weniger Zeit zwischen dem Auftreten und Erkennen einer Störung.

Nachteile der Detecting-Methode. Die Detecting-Methode mittels Agenten kann bei kleinen verteilten IT-Systemen mit wenigen Kontenpunkten einen Overhead beim Netzwerk-Traffic produzieren (Du et al., 2003). Der Einsatz von Agenten bedeutet zudem auch einen organisatorischen und technischen Aufwand für das Installieren und Bereithalten der Agenten, gerade mit dem Einsatz von vielen Plattformen, Applikationen und Systemen (Kufel, 2016). Sind Komponenten von Dritten Teil des verteilten IT-Systems ist außerdem fraglich, ob es möglich ist, Agenten auf diesen zu installieren.

4.4.4 Monitoring-Methode

Monitoring ist die Überwachung eines Systems mit geeigneten Mitteln, wozu auch Logging, Tracing und Detecting zählen können. Zum Monitoring gehört die Datensammlung, die Datenverarbeitung, die Datenaggregation und die Datenvisualisierung (Ewaschuk, 2016). Monitoring ist eine sehr weit verbreitete Methode in verteilten IT-Systemen geworden (Kufel, 2016). Störungen lassen sich damit gut erkennen. Dieser Methode sind wenige Grenzen gesetzt, was überwacht werden soll. Dazu gehören beispielsweise Anzahl von Fehlern, die Dauer von Verarbeitungsprozessen, die Lebensdauer von Servern oder auch die Anzahl von Anfragen (Ewaschuk, 2016). Monitoring ist daher nicht nur ein Werkzeug der Störungserkennung oder der Ursachenerkennung, sondern dient auch dazu, technische und fachliche Kennzahlen über das System bzw. den Geschäftsprozess zu erfassen.

Wie Abbildung 7 zeigt, ist die Monitoring-Methode außerhalb des Systems. Die anderen Methoden, Logging, Tracing und Detecting, sind Methoden, die innerhalb des Systems interagieren. Die Monitoring-Methode dagegen agiert extern und bedient sich den internen Methoden. So werden Ergebnisse vom Logging oder Tracing dem Monitoring zur Verfügung gestellt.

Möglichkeiten des Monitorings. Mit der Monitoring-Methode werden in der Praxis für die Störungserkennung besonders drei Bereiche in den Vordergrund gestellt, die überwacht werden. Dazu gehören die Verfügbarkeit, die Kapazität & Leistung und die Sicherheit. Im Folgenden sind mögliche Inhalte der drei Bereiche aufgelistet (Kufel, 2016):

- **Verfügbarkeit**
  - Von Systemen, Services oder Anwendungen
  - Gegenüberstellung von lauffähigen und nicht lauffähigen Komponenten
- **Kapazität & Leistung**
  - Leistung der System-Komponenten
  - Kapazität, die im System zur Verfügung steht
  - CPU- und Speicher-Auslastung oder Netzwerkbandbreite
- **Sicherheit**
  - Überwachung von Firewall-Logs, Verifikationen
  - Festhalten von Nicht-Autorisierten versuchten Zugriffen
  - Festhalten von misslungenen Passwort-Eingaben

Vorteile der Monitoring-Methode. Der wesentliche Vorteil dieser Methode ist es, einen umfassenden Überblick über das verteilte IT-System geben zu können (Kufel, 2016). Besonders Dashboards spielen in diesem Zusammenhang eine sehr wichtige Rolle. Hier können relevante Informationen schnell und klar dargestellt werden und verschaffen einen aktuellen Überblick über das Gesamtsystem (Ewaschuk, 2016). Abbildung 10 zeigt ein mögliches Dashboard für ein System. Besonders für das Management und auch für verantwortliche Personen, die Entscheidungen treffen müssen, sind solche Dashboards eine besondere Hilfe. Auch können Alerts und Warnungen visuell angezeigt werden (Kufel, 2016). Neben der Erkennung von Trends lassen sich auch zeitliche Vergleiche zwischen Messwerten einer alten und einer neu eingesetzten Technologie umsetzen.

Eine neu integrierte Datenbank kann beispielsweise mit Hilfe des Monitorings untersucht werden, um zu ermitteln, inwieweit diese performanter ist. Ein Messwert dafür wäre die durchschnittliche Antwortzeit.

Abbildung 10: Dashboards eines Monitoring-Systems (Bitos, 2020)

4.5 Techniken für die Ursachenerkennung

Neben den im vorherigen Kapitel 4.4 beschriebenen Methoden für die Ursachenerkennung beschreibt nun dieses Kapitel die Techniken für die Ursachenerkennung (Kavulya et al.). Zu den Techniken gehören:

Technik	Beschreibung
Regeln	Überwachung basiert auf vordefinierten Regeln und Expertenwissen.
Statistik	Untersuchung empirischer Daten auf Korrelationen und Vergleiche. Verwendung von Wahrscheinlichkeiten.
Modellierung	Modell des Systems als Basis, um Soll- und Ist-Zustand des Systems zu vergleichen.
Machine Learning	Verhaltensmustererkennung durch Erlernen des bisherigen Verhaltens auf Basis historischer Daten, darauf aufbauend Anomalie-Erkennung und Prognose.
Visualisierung	Mittels graphischer Darstellung von Trends können Erkenntnisse über Fehlerursachen gewonnen werden.

Tabelle 8: Übersicht über die Techniken für die Störungserkennung (Soila P. Kavulya et al.)

Diese Techniken werden in der Praxis bei den vorgestellten Methoden kombiniert. So wird z. B. für die Logging-Methode oft Machine Learning und Statistik eingesetzt.

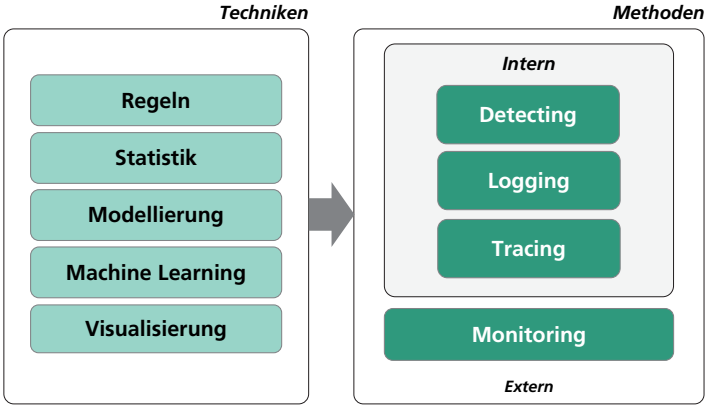

Abbildung 11: Techniken bei den verschiedenen Methoden zur Ursachenerkennung

Die aufgeführte Abbildung 11 zeigt nochmals einen Überblick über die Techniken und Methoden. Im Folgenden werden die einzelnen Techniken zur Ursachenerkennung beschrieben.

4.5.1 Regeln

Die regelbasierte Technik verwendet im Kern vordefinierte Regeln und vorhandenes Expertenwissen. Über vordefinierte Regeln werden Probleme festgehalten, um Ursachen zu diagnostizieren. Eine bekannte Methode für die Definition eines Regelwerks ist das Codebook, bei dem in einer Matrix zu bestimmten Störungen Symptome festgehalten werden, wie Abbildung 12 zeigt.

Für die Regeln werden sogenannte Forward-chaining Inference Mechanisms (Vorwärts-verkettete Mechanismen zur Ableitung) angewandt. Diese setzen sich zusammen aus einem *Faktum* und einer *Folgerung*: *Wenn-Part* (Annahme) und einem *Dann-Part* (Schlussfolgerung). Dies wäre beispielsweise: Wenn CPU-Auslastung > 90%, dann ist Server überlastet.

		Störung									
		ST1	ST2	ST3	ST4	ST5	ST6	ST7	ST8	ST9	ST10
Symptome	S1	x			x				x		
	S2										
	S3		x	x			x				
	S4				x				x	x	
	S5	x				x					x

Abbildung 12: Aufbau eines Codebooks mit Zuordnung von Symptomen zu Störungen

Bei dieser Technik sind jedoch auch einige Einschränkungen zu berücksichtigen. Die Regeln basieren grundsätzlich auf einer subjektiven Einschätzung. Daher ist es wichtig, Experten einzusetzen, um Aussagen treffen zu können. Darüber hinaus verkörpern solche Regeln komplexe Wissenszusammenhänge, die nicht einfach zu visualisieren sind.

4.5.2 Statistik

Statistik ist eine sehr breit eingesetzte Technik für die Untersuchung von empirischen Daten. Diese Daten werden gesammelt, analysiert und interpretiert. Mit Hilfe von Korrelationen, Vergleichen von Histogrammen und Wahrscheinlichkeiten ist es dann möglich, auf vorhandene Störungen zu schließen. So kann als Beispiel die Korrelation zwischen dem Datendurchsatz und dem der CPU-Auslastung untersucht werden und eine Wahrscheinlichkeit von Datenbankstörungen ermittelt werden.

Bekannte Methoden. Häufig eingesetzte Methoden in der Statistik sind das *parametrisierte Verfahren* und das *nicht-parametrisierte Verfahren*. Das *parametrisierte Verfahren* ist ein überwachtes Verfahren, bei dem eine Modellstruktur schon vorgegeben ist, beispielsweise bei einer linearen und nichtlinearen Regression. Das *nicht-parametrisierte Verfahren* ist nicht überwacht und es ist keine Modellstruktur bekannt. Hier werden oft Entscheidungsbäume eingesetzt.

Vor- und Nachteile. Während die statistischen Techniken wenig Expertenwissen über Systemdetails brauchen und auch die Diagnosetechniken auf fundierten mathematischen Grundlagen basieren, müssen natürlich die Daten ausreichend und aussagekräftig sein, um signifikante Aussagen treffen zu können. Darüber hinaus müssen natürlich auch Experten für Auswertungen herangezogen werden, denn semantische Auswertungen können statistische Techniken nicht abbilden.

4.5.3 Modellierung

Bei dieser modellbasierten Technik wird aus dem realen System ein Modell definiert (modelliert). Der aktuelle Stand des Systems wird dann mit dem Modell verglichen, um Probleme und deren Ursachen zu erkennen. Da ein Modell immer nur eine begrenzte Sicht des realen Systems darstellt, muss der Fokus des Modells auf einen spezifischen Kontext gelegt werden. Dies kann ein Kommunikationsmodell sein, das den Aufbau der Netzwerkinfrastruktur beschreibt, ohne die innere Logik der Komponenten abzubilden.

Diese modellbasierte Technik wird weiter in drei bekannte Arten unterteilt: die *physische modellbasierte Technik (1), das Regressions- bzw. Warteschlangenmodell (2)* und das *Graphen-theoretische Modell (3).*

Physische modellbasierte Technik (1). Bei dieser Technik wird das verteilte IT-System physisch im Labor nachgebaut. Es handelt sich hierbei um ein geschlossenes System. Wie Abbildung 13 zeigt wird aus dem realen verteilten IT-System ein spezifisches Gebiet herausgenommen und dieses dann in einem Modell abgebildet. Mit diesem Modell wird dann versucht, entsprechende in der Realität stattgefundene Probleme zu rekonstruieren. So können dann Ursachen im »Labor« erkannt werden.

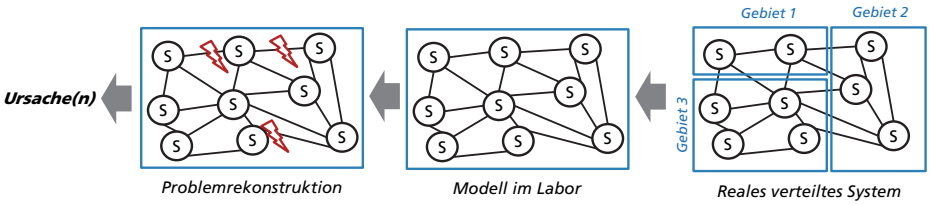

Abbildung 13: Physische modellbasierte Technik: Reale Welt in ein physisches Modell

Regressions- bzw. Warteschlangenmodell (2). Diese Art von Modell beschäftigt sich mit der Ursachenerkennung von Workload-, Kapazitäts-, und Performance-Problemen. Auch wird es für Korrelationen zwischen Ressourcenverbrauch und Anwendungsverhalten eingesetzt, wofür die *Mean Value Analysis* eingesetzt wird. Diese Analyse analysiert Durchschnittswerte von unterschiedlichen Use Cases.

Abbildung 14 zeigt eine solche Analyse, die die Antwortzeiten von Transaktionen in einem verteilten IT-System untersucht. Dazu werden Transaktionen in einer Queue in das System verteilt (gesendet). Mit dem Verschicken bzw. dem Verarbeiten der einzelnen Transaktionen kann das Verhalten der einzelnen Komponenten untersucht werden. So können Anomalien erkannt werden, mittels der Auswertung von Antwortzeiten.

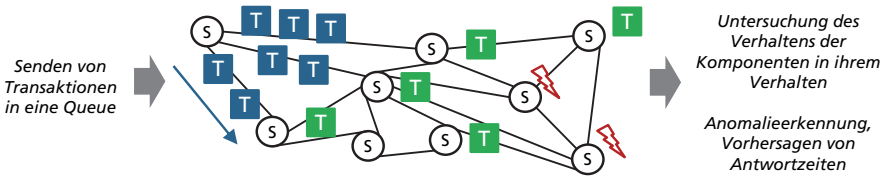

Abbildung 14: Mean Value Analysis: Untersuchung der Antwortzeiten von Transaktionen

Graphen-theoretisches Modell (3). Beim Graphen-theoretischem Modell werden Graphen aufgebaut, um Beziehungen zwischen Komponenten oder auch Muster im System zu modellieren. Beispielsweise findet dieses Modell häufig Anwendung, um Kommunikationsmuster zu untersuchen (z. B. Netzwerkkommunikation). Hier kann dann eine erfolgreiche Kommunikation zwischen Komponenten und Kommunikationsstörungen im Kommunikationsmuster diagnostiziert werden als auch Wahrscheinlichkeiten von einer erfolgreichen Kommunikation und Kommunikationsstörungen.

Vorteile & Einschränkungen. Diese Methode ist geeignet für die Diagnose der Kommunikation zwischen Applikationen. Darüber hinaus kann auch das Verhalten von Semantik von Komponenten miteinbezogen werden (z. B. Veränderungen des Kommunikationsverhaltens bei sehr hohem Datenverkehr). So verursacht eine Störung unrechtmäßige Workload-Veränderungen, womit unterschieden werden kann, ob ein Verhalten legitim oder nicht-legitim war. Jedoch ist für diese Methode ein tiefes Verständnis des Systemverhaltens notwendig, um das Systemmodell zu definieren. Auch kann eine Semantik nicht automatisch abgebildet werden. Bezüglich des Erkennens von Störungen können nur solche erkannt werden, die im Modell definiert sind (neue Probleme müssen im Modell abgebildet werden).

4.5.4 Machine Learning

Das Maschine Learning ist ein Teilbereich der Künstlichen Intelligenz. Hier wird versucht, aus historischen Daten automatisch ein Modell abzuleiten, um Voraussagen treffen zu können. Für das Lernen werden sogenannte Trainingsdaten verwendet, zum Beispiel reale Messdaten für Zeiträume der normalen Funktionalität und der Störung. Anhand dieser Trainingsdaten lernt die Künstliche Intelligenz das Verhalten des Systems. So können Störungen schon vorausgesagt und Ursachen schneller gefunden werden.

Bekannte Methoden. Zu den bekannten Methoden gehört das supervised und das unsupervised Learning. Ersteres benutzt Trainingsdaten mit schon vorhandenen Ergebnissen, also beispielsweise, ob eine Störung vorlag oder nicht (z. B. künstliche neuronale Netze). Bei der zweiten Methode werden Daten ohne Vorarbeit (labeling) analysiert, sodass die Ergebnisse von vorneherein nicht bekannt sind. Besonders die Cluster-Methode wird hier häufig als Werkzeug eingesetzt, beispielsweise, um typische Systemzustände zu identifizieren.

Vorteile & Einschränkungen. Machine Learning ist eine neue und vielversprechende Technik. Es ist möglich, Systemverhalten zu lernen beispielsweise mittels Clustering von bekannten Fehlverhalten für Ursachenerkennung. Jedoch ist zu berücksichtigen, dass Ergebnisse immer auf Wahrscheinlichkeiten beruhen. Die Qualität der Ergebnisse hängt außerdem stark mit der Qualität und Quantität der Trainingsdaten zusammen. Verändert sich ein verteiltes IT-System häufig, ist Maschine Learning schwieriger einsetzbar, weil erneutes Lernen notwendig ist, wenn sich das verteilte IT-System verändert.

4.5.5 Visualisierung

Bei der Visualisierung geht es grundsätzlich um die geeignete Sichtbarmachung von Informationen über ein verteiltes IT-System. Dazu gehört auch das Sichtbarmachen von Trends (z. B. durch Diagramme), die Visualisierung von interaktiven Graphen, um Hypothesen für Ursachenerkennung zu untersuchen. Auch Werte aus Diagnosetools können visualisiert werden (z. B. Netzbelastung oder Anzahl der offenen Kundenanfragen). Gerne werden auch Warn-Detektoren im Monitoring integriert, um entsprechend reagieren zu können.

Dashboards spielen für die Visualisierung eine besondere Rolle. Sie bieten die Möglichkeit, relevante Daten zu nützlichen Informationen aggregiert und visuell aufbereitet anzuzeigen. Dabei ist die Konfiguration des Dashboards zielgruppenspezifisch. Ein IT-Manager braucht andere Kennzahlen als ein fachlicher Prozessverantwortlicher.

Vorteile & Einschränkungen. Durch Dashboards können Trends mittels Expertenwissen und Beobachtungen logischer Zusammenhänge erkannt werden, die auf Ursachen zurückschließen lassen. Jedoch ist die Visualisierung immer von anderen Methoden (um Daten zu verarbeiten) abhängig, wie beispielsweise von Machine Learning oder anderen Tools, die Daten verarbeiten. Darüber hinaus werden auch nur Ergebnisse und Symptome gezeigt und keine Ursachen. Diese müssen erst semantisch hergeleitet werden. Auch wenn Dashboards über Mechanismen zum Alerting verfügen, sind sie grundlegend Werkzeuge zum aktiven Monitoring, benötigen also personellen Aufwand.

5 Stand der Forschung und Technik

5.1 Aktuelle Anwendungslösungen

Auf dem Markt existieren eine Vielzahl von Technologien und Softwarelösungen zur Überwachung und Störungserkennung. Im Zuge der steigenden Adoption von Cloud-Lösungen und der immer komplexeren verteilten IT-Systeme entwickeln sich auch die Softwarelösungen weiter.

Im Folgenden werden die wichtigsten Technologien beschrieben. Exemplarisch werden einzelne verbreitete Softwareprodukte vorgestellt, um einen Überblick über den Markt zu geben. Eine vollständige Marktstudie in allen Technologiekategorien würde den Umfang dieser Übersichtsstudie sprengen. Wo möglich wird auf weiterführende Quellen verwiesen. Tabelle 9 gibt eine Übersicht der vorgestellten Technologien zur Überwachung und Störungserkennung.

Technologie	Beschreibung
Simple Network Management Protocol *(SNMP)*	Protokoll zur Überwachung und Steuerung von IT-Komponenten.
IT-Infrastruktur-Überwachung	Überwachung von IT-Infrastruktur, Netzwerken und Systemen.
Log-Analyse	Auswertung von Log-Dateien in verteilten IT-Systemen.
Application Performance Management *(APM)*	Überwachung von Anwendungen und Systemen.
Stream Analytics	Überwachung mittels Strömen von Ereignissen.
Business Intelligence	Gewinnung von Wissen aus Daten.
Business Activity Monitoring / Process Monitoring	Überwachung von Geschäftsprozessen.

Tabelle 9: Übersicht über die Technologien zur Überwachung und Störungserkennung

5.1.1 Technologienklassifikation zur Störungserkennung

Die am Markt verfügbaren Technologien zur Störungserkennung und Überwachung unterscheiden sich gemäß wichtiger Merkmale und Einsatzzwecke. Bei der Technologieauswahl ist es daher wichtig, zu wissen, wie sich die fachlichen und technischen Anforderungen auf die einzelnen Merkmale auswirken. Entscheidend für die Auswahl der richtigen Merkmale sind die Anforderungen an die Technologien. Konkretisieren lassen sich diese, indem spezifiziert wird, welche Fragen eine Softwarelösung beantworten muss, bzw. welche Daten diese liefern muss, z. B.:

- *Erkennung aktueller Fehler in jedem Einzelfall?*
- *Statistische Werte des Gesamtsystems?*
- *Rückblickende Diagnose und Ursachenerkennung?*

Auch müssen Beschränkungen des zu überwachenden Systems beachtet werden, z. B.:

- *Ist eine Kommunikation in Echtzeit möglich?*
- *Welche Daten stehen in den Systemen zur Verfügung?*
- *Welcher Zugriff auf die Systeme ist möglich?*

Die im Folgenden vorgestellten Technologien sind nach verschiedenen, auf diesen Fragen basierten Merkmalen klassifiziert, um die Auswahl zu erleichtern.

Lösungen überwachen verschiedene Objekte (Abbildung 15). Auf der untersten Ebene werden IT-Komponenten technisch überwacht. Auf der obersten Ebene werden Prozesse fachlich überwacht. Die Überwachung von Systemen liegt zwischen den Ebenen. Üblich ist hier eine Mischung von fachlicher und technischer Überwachung.

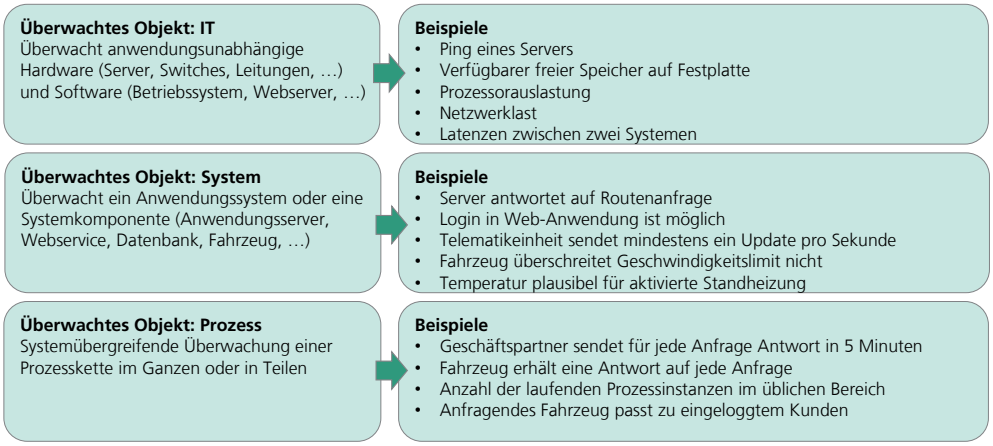

Abbildung 15: Klassifikation nach überwachtem Objekt

Jedes Überwachungswerkzeug benötigt Daten, anhand denen es die Überwachung durchführt. Die Einzelheiten zeigt Abbildung 16. Als »pull« bezeichnet man es, wenn das Werkzeug aktiv Daten aus dem überwachten Objekt einholt. Das Gegenstück ist »push«, bei dem das überwachte Objekt Daten an das Überwachungswerkzeug schickt. Polling ist eine Mischform. Wie die Daten übermittelt werden, hängt von den technischen Gegebenheiten ab, denn beide Arten des Transfers haben Vor- und Nachteile.

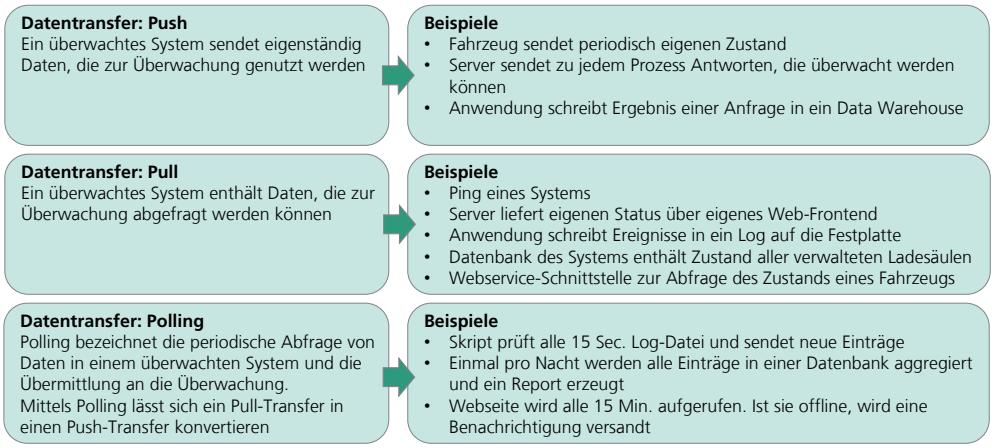

Abbildung 16: Klassifikation nach Datentransfer

Das Betrachtungsprinzip ist die grundlegende Art und Weise, nach der ein Überwachungswerkzeug arbeitet (Abbildung 17). Datenbasierte Überwachung betrachtet im System vorhandene Daten, z. B. Datenbankeinträge, und wertet auf diesen Regeln aus. Daten sind zunächst nicht zeitgebunden, können sich aber über die Zeit ändern. Man betrachtet also das, was »ist«. Im Vergleich dazu betrachtet die nachrichtenbasierte Überwachung die im System auftretenden Nachrichten. Diese sind an einen Zeitpunkt gebunden, und ändern sich nicht, nachdem sie einmal erstellt wurden. Es wird also betrachtet, was »geschieht«.

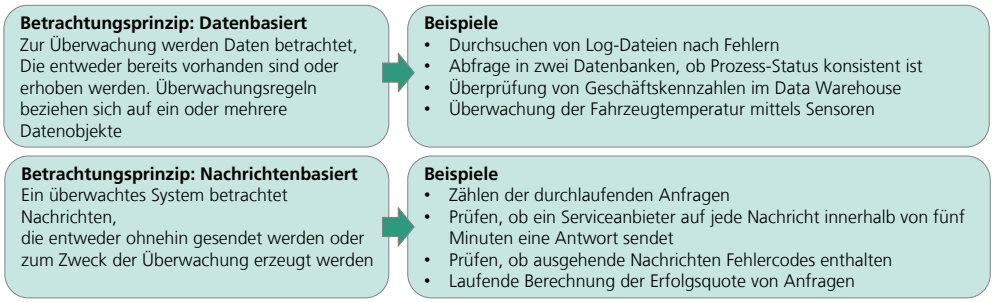

Abbildung 17: Klassifikation nach Betrachtungsprinzip

Der Betrachtungszeitraum beschreibt, in welchen Zeiträumen die Überwachung funktioniert (Abbildung 18). Überwachung in Echtzeit fokussiert sich auf das, was gerade geschieht (und die sehr nahe Vergangenheit). Der Fokus liegt hier darauf, den aktuellen Systemzustand zu erfassen und aktuelle Störungen zu erkennen. Historische Überwachung betrachtet längere Zeiträume, z. B. Wochen oder Monate. Hier liegt der Fokus darauf, die langfristige Entwicklung des Systemzustands zu erkennen und anormales Verhalten zu erkennen oder sogar vorauszusagen.

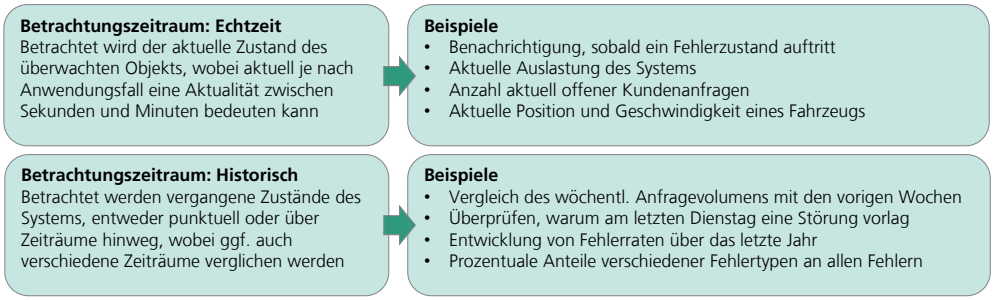

Abbildung 18: Klassifikation nach Beobachtungszeitraum

Der Betrachtungsumfang legt fest, in welcher Granularität die Überwachung arbeitet (Abbildung 19). Globale Überwachung betrachtet das System als Ganzes, um Dinge wie Auslastung und Durchschnittszeiten zu messen. Überwachung einer Instanz betrachtet im Gegensatz den Einzelfall, um Dinge wie Service-Level und Ausnahmefehler zu messen.

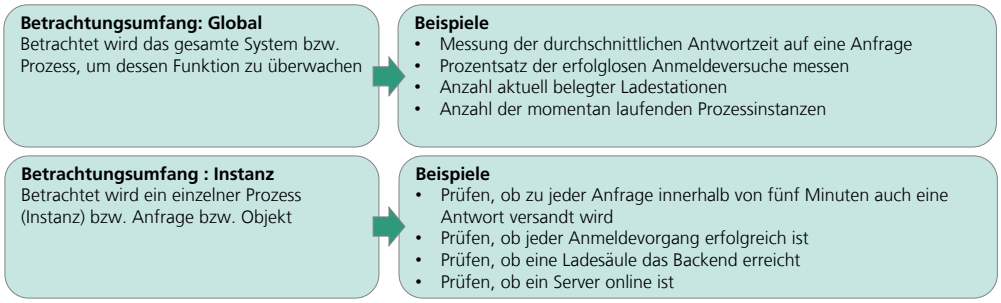

Abbildung 19: Klassifikation nach Betrachtungsumfang

Zu allen Klassifikationsmerkmalen ist anzumerken, dass diese meistens nicht exklusiv sind. Das heißt, in vielen Lösungen gibt es Mischformen, die Merkmalskombinationen unterstützen. Welche Ausprägungen geeignet sind, hängt von den Anforderungen an die Überwachung und Störungserkennung ab.

Prozess	Instanz	Global
System	Echtzeit	Historisch
IT	Push	Pull
Datenbasiert	Nachrichtenbasiert	

5.1.2 Simple Network Management Protocol

Das Simple Network Management Protocol (SNMP) ist ein von der IETF entwickeltes standardisiertes Protokoll zur Überwachung und Steuerung von IT-Komponenten, die per IP erreichbar sind. Der Fokus liegt hier insofern nicht nur auf Rechnern oder Servern, sondern auch auf Netzwerkhardware wie Switches und Routern.

In Abbildung 20 sind die grundlegenden Komponenten einer SNMP-basierten Lösung aufgezeigt. Ein SNMP-Manager dient als Überwachungswerkzeug bzw. Komponente eines Überwachungswerkzeugs. Mittels GET-Abfrage können aktuelle Kennwerte eines SNMP-Agent abgefragt werden. IT-Komponenten, die SNMP unterstützten, beinhalten einen solchen Agenten. Mögliche Werte, die abgefragt werden können, sind beispielsweise Up-Time, CPU-Temperatur, Auslastung, Fehlerzähler und Festplattenplatz. Eigene Abfragen und Werte können definiert werden. Mittels Set-Anfragen können auch Werte gesetzt werden, z. B. um Konfigurationsparameter zu ändern. SNMP-Manager können auch Nachrichten von Komponenten empfangen, beispielsweise Fehlermeldungen. Diese Nachrichten werden als TRAP bezeichnet und enthalten einen Nachrichtentext sowie eine beliebige Anzahl an Kennwerten. (Zobel and Behnsen, 2019)

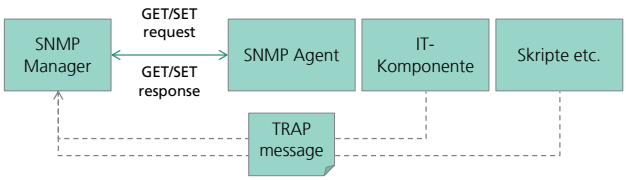

Abbildung 20: Komponenten und Abläufe von SNMP

Zur Identifikation von Kennwerten werden in SNMP sogenannte Object Identifiers (OIDs) verwendet, die ähnlich einer IP-Adresse aus Zahlen und Punkten bestehen. OIDs sind nicht menschenlesbar, werden aber mittels sogenannter Management Information Databases (MIBs) dokumentiert. Standardisierte OIDs für Betriebssysteme, Router o.Ä. existieren. Hardware- und Software-Hersteller haben ein eigenes, eindeutiges Segment in der OID-Hierarchie und können dort eigene Werte definieren. Eigene OIDs können unter einem Hersteller-Prefix hinterlegt werden. (DPM Telecom, 2019)

	Inhalt
Vorteile	*+ Standardisiert* *+ Breite Unterstützung durch IT-Komponenten, Betriebssysteme und Software-Bibliotheken* *+ Systematische Kodierung durch OIDs* *+ Erweiterbar durch eigene OIDs* *+ Einfache Abdeckung von Standard-Überwachungsaufgaben* *+ Sowohl Datenabfragen als auch Nachrichtentransfer möglich*
Nachteile	*- Entgegen des Namens nicht einfach zu implementieren* *- Nachrichtentransfer mittels fire-and-forget – hoher Aufwand zur Fehlersuche bei Einrichtung* *- Sicherheit der Datenübertragung erst seit Version 3 berücksichtigt*

Tabelle 10: Vor- und Nachteile des Simple Network Management Protocol

5.1.3 Stream Analytics

Prozess	Instanz	Global
System	Echtzeit	Historisch
IT	Push	Pull
Datenbasiert	Nachrichtenbasiert	

Stream Analytics bezeichnet Technologien zur Auswertung von Ereignisströmen. Ereignisse (engl. Events) bestehen aus einem Zeitpunkt sowie zugehörigen Daten. Ein Beispiel ist der Beginn eines Prozesses mit Zeitpunkt, Prozess-ID und Bearbeiter. Auch Nachrichten in einem vernetzten IT-System sind Ereignisse.

Im grundlegenden Fall werden auftretende Ereignisse im System erkannt, verarbeitet und falls nötig wird auf Ereignisse mit einem Folgeprozess reagiert, wie in Abbildung 21 dargestellt (Bruns and Dunkel, 2010).

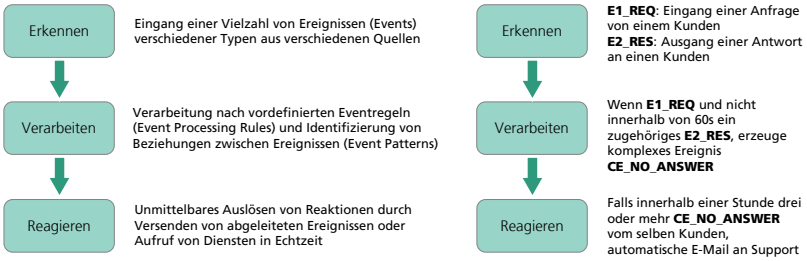

Abbildung 21: Ablauf der Ereignisverarbeitung

Der Verarbeitung liegen sogenannte Verarbeitungsregeln zugrunde. Diese Verarbeitungsregeln erkennen Muster im Ereignisstrom und werten diese aus. Beispiele für solche Muster sind in Abbildung 22 gezeigt.

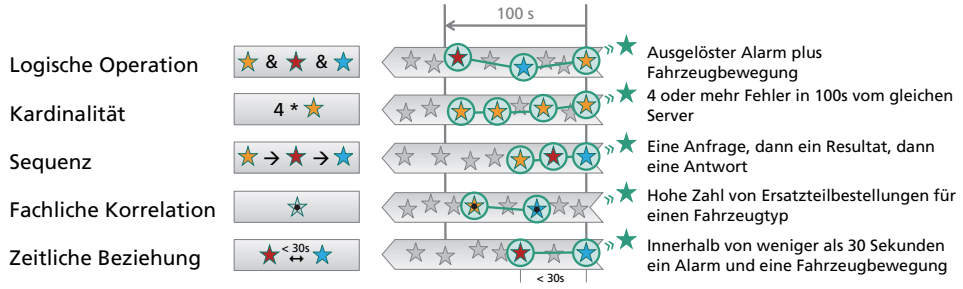

Abbildung 22: Beispiele für Ereignismuster

Als Begrenzung dafür, was als Ereignismuster zählt, werden sogenannte Fenster (engl. windows) verwendet. So kann ein Muster beispielsweise auf eine Zeitspanne von 60 Sekunden oder auf eine Anzahl von 1000 Ereignissen begrenzt werden. Typischerweise beziehen sich Ereignismuster auf einzelne Vorgänge oder Prozessinstanzen.

Auf solchen quantitativen oder zeitlichen Fenstern lassen sich auch aggregierte Werte berechnen, mit denen Aussagen über das Gesamtsystem gemacht werden können. Beispiele für Aggregationen sind:

Aggregation	Beispiel
Anzahl	*Anzahl der erfolgreichen Anfragen in der letzten Stunde.*
Durchschnitt	*Durchschnittliche Antwortzeit über die letzten 50 Anfragen.*
Summe	*Gesamtumsatz über alle Bestellungen der letzten 24 Stunden.*
Max/Min:	*Maximale Antwortzeit über die letzten 10 Anfragen.*

Tabelle 11: Beispiele für Aggregationen

Aggregationen sind besonders dafür geeignet, Key Performance Indikatoren in Echtzeit zu berechnen. Es ist ebenfalls möglich, mehrere Werte in einer Formel zu kombinieren, solange zur Betrachtung das gleiche Fenster gewählt ist. Ebenso ist es möglich, betrachtete Ereignisse zu filtern, z. B., um nur Anfragen mit Fehlerzustand zu betrachten.

Typische Lösungen zur Ereignisverarbeitung (siehe Abbildung 23) können eine Vielzahl von Ereignisquellen anbinden. Ereignisse werden in Echtzeit nach definierten Mustern und Regeln verarbeitet. Das Resultat der Verarbeitung ist die Erzeugung neuer Ereignisse. Sogenannte komplexe Ereignisse (engl. complex events) sind Ereignisse, die sich aus mehreren anderen Ereignissen zusammensetzen. Neu erzeugte Ereignisse können wiederum als Eingangs-Ereignisse für die Verarbeitung dienen, sodass eine mehrstufige Verarbeitung möglich ist. Nachgelagerte Komponenten sind Ereignissenken, die bei für sie relevanten Ereignissen benachrichtigt werden und Folgeprozesse auslösen können (z. B. E-Mail-Benachrichtigung, Anzeige auf Dashboard). In der Vergangenheit wurde statt des neuen Begriffs Stream Processing auch der Begriff Complex Event Processing (CEP) verwendet. Es stellt sich die Frage, ob es sich hierbei um eine neue Technologie oder doch nur neue Terminologie, also »alten Wein in neuen Schläuchen« handelt.

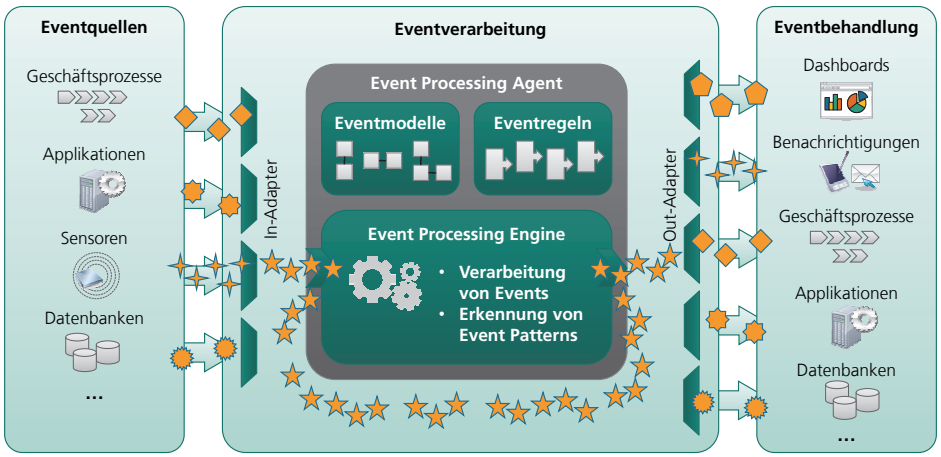

Abbildung 23: Architektur für Ereignisverarbeitung (nach (Bruns and Dunkel, 2010))

In den letzten Jahren haben sowohl IBM als auch Oracle ihre Produkte von CEP in Stream Analytics umbenannt. TIBCO verwendet sogar beide Begriffe parallel. Eine mögliche Erklärung ist, dass keine Marketingabteilung ein Produkt haben will, das »komplex« ist, sondern lieber eines, das einfach ist. Mit Blick in die Broschüren der Hersteller werden oft Unterschiede zwischen Stream Processing und Complex Event Processing genannt:

- Stream Processing ist dezentral und somit besser skalierbar
- Stream Processing verwendet normale Programmiersprachen statt Spezialsprachen für die Definition von Regeln und Mustern
- Stream Processing ist Big-Data-tauglich, hat dafür aber höhere Latenzen

Diese Punkte gelten allerdings auch für die aktuellen Versionen vieler CEP-Technologien. In dieser Studie wurde daher der modernere Begriff »Stream Processing« verwendet. Folgende Tabelle 12 zeigt die Vor- und Nachteile des Stream Processings:

	Inhalt
Vorteile	*+ Flexibel und skalierbar* *+ Erkennen von Störungen über Systemgrenzen hinweg* *+ Betrachtung einzelner Anfragen und des Gesamtsystems* *+ Einfache Visualisierung*
Nachteile	*- Keine Out-Of-The-Box-Lösung* *- Expertenwissen für Regelerstellung notwendig* *- Aufwändige Integration in Bestandssysteme* *- Kontinuierliches Finetuning der Regeln*

Tabelle 12: Vor- und Nachteile des Stream Processings

5.1.3.1 Technologie: Esper

Esper ist die führende Open Source CEP-Engine (siehe Abbildung 24). Folgende Eigenschaften sind dieser Technologie zuzuordnen:

Name: Esper
Art: Complex Event Processing
Firma: EsperTech (US)
Lizenz: OpenSource/Kostenpflichtig
Web: espertech.com/esper/

- ✓ Kein eigenes Werkzeug, sondern Komponente
  - ➢ Integration in eigene Software mittels Adaptern
  - ➢ Java, XML, .Net, CSV, JMS, Datenbank, Socket oder HTTP
  - ➢ Erweiterbar mittels Plugins
- ✓ Event Processing Language (EPL) zur Definition von Regeln
  - ➢ Unterstützt alle genannten Patterns und Aggregationen
- ✓ Hochperformant im Vergleich zu anderen Engines
- ✓ In der kommerziellen Variante
  - ➢ Skalierbar durch multiple Instanzen
  - ➢ Grafisches Frontend zur Administration
  - ➢ Entwicklungswerkzeug für Ereignisregeln

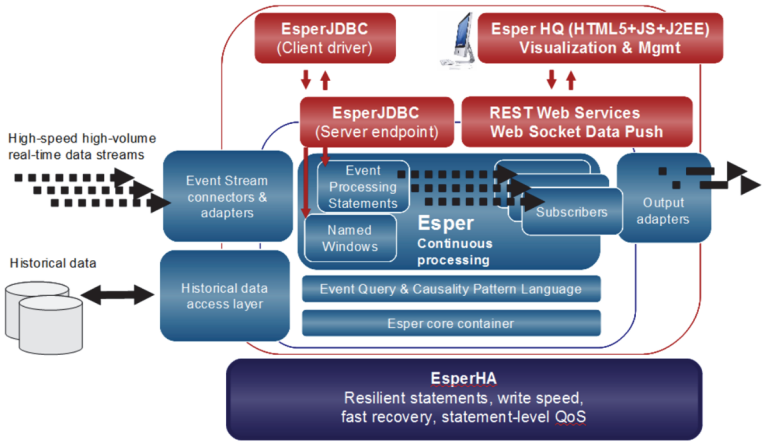

Abbildung 24: Komponenten von Esper (EsperTech)

5.1.3.2 Technologie: TIBCO StreamBase

TIBCO StreamBase ist eine Complex Event Processing Plattform. Folgende Eigenschaften sind dieser Technologie zuzuordnen:

Name: TIBCO StreamBase
Art: Complex Event Processing
Firma: TIBCO Software (US)
Lizenz: Kostenpflichtig
Web: tibco.com

- ✓ Regelerstellung grafisch und durch Programmierung
- ✓ Eclipse-basierte Entwicklungsumgebung für Analytics
- ✓ Integration von Datenquellen mittels 150+ Adaptern
- ✓ Unterstützung für IoT-Anwendungen
- ✓ Dashboard-Lösung enthalten
- ✓ Definition von Reaktionen wie Alerts und Folgeprozesse
- ✓ Analytics mittels Programmiersprache R
- ✓ Hochverfügbar und skalierbar
- ✓ Data-Warehouse-Komponente Datamart

5.1.3.3 Technologie: Apama Streaming Analytics

Apama ist eine Complex Event Processing-Technologie und ein Event Stream Processing Engine. Folgende Eigenschaften sind dieser Technologie zuzuordnen:

- ✓ Apama kann Streaming-Analysen in Großen verteilten IT-System durchführen
- ✓ Es lassen sich Datenmuster definieren, die überwacht werden
- ✓ Apama ist spezialisiert auf einen hohen Datendurchsatz, eine niedrige Latenz und eine effiziente Speicherperformanz
- ✓ Apama verwendet die Event Processing Language (EPL)
- ✓ Apama bietet eine in-build Technologie für Dashboards an

Name: Apama
Art: Complex Event Processing
Firma: Software AG (DE)
Lizenz: Kostenpflichtig
Web: softwareag.com/de/product/data_analystics/analytics/default.html

5.1.3.4 Technologie: Oracle Stream Analytics

Oracle Stream Analytics ermöglicht den Anwendern eine große Menge an Echtzeit-Informationen mittels Machine Learning oder Korrelations-Mustern zur verarbeiten (Abbildung 25). Folgende Eigenschaften sind dieser Technologie zuzuordnen:

- ✓ CEP-Engine mit Unterstützung für Geodaten
- ✓ Integration mit Oracle Datenbanken
- ✓ Verwendung von Streaming-Expression und Business-Rules-Analysen
- ✓ Unterschiedliche Perspektiven für verschiedene Industrien
- ✓ Verwendung von statistischen Auswertungen, Stream Anomalie-Erkennung und Streaming Machine Learning
- ✓ Graphische Visualisierung von Streaming Information
- ✓ Bereitstellen von Aktionen, um auf Analysen reagieren zu können

Name: Oracle Stream Analytics
Art: Complex Event Processing
Firma: Oracle (US)
Lizenz: Kostenpflichtig
Web: oracle.com/middleware/technoloies/stream-processing.htm

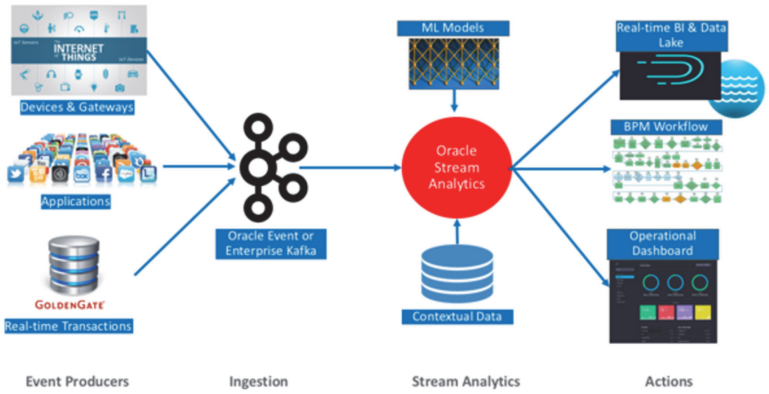

Abbildung 25: Übersicht über die Architektur der Oracle Stream Analytics (Oracle, 2019b)[4]

[4] Copyright Oracle and its affiliates. Used with permission.

5.1.3.5 Technologie: IBM Streaming Analytics

IBM Streaming Analytics ist eine Complex Event Process Engine. Diese kann in ein IBM-Ökosystem integriert werden. Diese Technologie ist dafür ausgelegt, eine große Menge an unstrukturierten Daten (z. B. Text, Video, Audio oder auch Sensordaten) zu verarbeiten (IBM, 2019). Folgende Eigenschaften sind dieser Technologie zuzuordnen:

Name: IBM Stream Analytics
Art: Complex Event Processing
Firma: IBM (US)
Lizenz: Kostenpflichtig
Web: ibm.com/doud/streaming-analytics

- ✓ Unterstützt Programmiersprachen wie Java, Scala oder Python
- ✓ Verwendung von integrierten Entwicklungsumgebungen mit visueller Oberfläche
- ✓ Anbindung von jeglichen Datenquellen
- ✓ Integration von Hadoop oder Spark
- ✓ Verwendung integrierten Analysetechniken wie Machine Learning
- ✓ Umsetzung von Applikationen mit schnellem Daten-Streaming
- ✓ Hilfe beim Optimieren der Rechenpower bei IoT-Applikationen

5.1.4 IT-Infrastruktur-Überwachung

Prozess | Instanz | Global
System | Echtzeit | Historisch
IT | Push | Pull
Datenbasiert | Nachrichtenbasiert

IT-Infrastruktur-Überwachungswerkzeuge dienen der Überwachung der technischen Funktionalität von IT-Infrastruktur wie Servern und Routern. Typische Prüfungen von IT-Systemen sind die Erreichbarkeit im Netzwerk (z. B. mittels Ping), Auslastung von Ressourcen wie Prozessoren und Netzwerk sowie vorhandener Haupt- und Festplattenspeicher. Auch die Überprüfung von Sicherheitsmerkmalen ist möglich, zum Beispiel die Gültigkeit von SSL-Zertifikaten oder die Erkennung unnormaler Nutzungsmuster. Vereinzelt werden mittels dieser Werkzeuge auch Prüfungen auf Systemebene durchgeführt, zum Beispiel durch den Probe-Aufruf von Webseiten oder das Durchführen eines Beispiel-Requests.

Prüfungen sind typischerweise einzeln und werden periodisch durchgeführt, z. B. alle 60 Sekunden. Aus vergangenen Prüfungen wird eine Historie aufgebaut, sodass die Entwicklung von Kennwerten über die Zeit eingesehen werden kann. Vereinzelt werden diese Historien schon benutzt, um anormales Verhalten zu erkennen (z. B. Anfragevolumen unüblich für sonntags) oder um Ausfälle zu prognostizieren (z. B. deuten beständig steigende Antwortzeiten auf das baldige Verletzen eines Service Level Agreements hin).

Weichen die gemessenen Ist-Werte von den Soll-Werten ab, können Alarme generiert werden, die auf Dashboards angezeigt oder per E-Mail, SMS etc. versandt werden. Ein Beispiel für solch einen Alarm ist in Abbildung 26 gezeigt.

Abbildung 26: E-Mail Alert für ein ablaufendes SSL-Zertifikat mit Historie in PRTG

IT-Infrastruktur-Überwachungswerkzeuge sind sehr ausgreift und unterstützen eine breite Anzahl an Prüfungen und Standards. Die Einrichtung für typische Infrastrukturen erfolgt weitgehend automatisch. Im Vergleich zu anderen Technologien ist nur ein geringer Anpassungsaufwand notwendig.

	Inhalt
Vorteile	*+ Out-Of-The-Box-Lösung* *+ Große Anzahl an vordefinierten Prüfungen* *+ Alarme* *+ Automatische Historie* *+ (Begrenzte) Überwachung von Systemen Dritter möglich*
Nachteile	*- Keine fachliche Prüfung der Funktionalität* *- Alerts und Dashboards für Nicht-Techniker ungeeignet*

Tabelle 13: Vor- und Nachteile von IT-Infrastruktur-Überwachung

5.1.4.1 Technologie: PRTG Network Monitor

Der PRTG Network Monitor (ehemals Paessler Router Traffic Grapher) ist ein Softwarewerkzeug zur Überwachung von IT-Systemen und Netzwerken. Überwacht werden Netzwerk-Traffic, VMs, Pings, Festplatten, Webseiten, etc.

- ✓ Prüfungen über das Netz (Pull) oder mittels lokaler Probes (Push)
  - ➢ Protokollunterstützung SNMP, SSH,
  - ➢ Ping, HTTP, SSL, REST, SQL, etc.

Name: PRTG
Art: IT-Infrastruktur-Überwachung
Firma: Paessler (DE)
Lizenz: Kostenpflichtig
Web: de.paessler.comprtg

- ✓ Alert-Funktion mittels App, SMS oder E-MailSehr hoher Funktionsumfang für gebräuchliche IT-Komponenten
- ✓ Erweiterbar mittels API und Scripting
- ✓ Verfügbar zum Selbst-Hosten oder als gehostete Cloud-Lösung
- ✓ Hochverfügbar mittels Clustering
- ✓ Skalierbar durch multiple Instanzen

5.1.4.2 Technologie: Nagios

Nagios ist ein Open-Source Softwarewerkzeug zur Überwachung von IT-Systemen und Netzwerken und gilt als die Open-Source-Alternative zu PRTG.

Name: Nagios
Art: IT-Infrastruktur-Überwachung
Firma: Nagios Enterprises (USA)
Lizenz: Open Source
Web: nagios.com / nagios.org

- ✓ Prüfungen über das Netz (Pull) oder mittels lokaler Agenten (Push)
- ✓ Alert-Funktion mittels E-Mail, andere wie SMS durch Erweiterungen
- ✓ Erweiterbar durch Plugin-Konzept und Scripting
- ✓ Zahlreiche Plugins für gebräuchliche IT-Komponenten
- ✓ Open Source mit optionalem kommerziellen Support
- ✓ Kommerzielle Variante bietet zusätzliche Convenience-Features
- ✓ Skalierbar durch Zusatzprodukt Nagios Fusion
- ✓ Zusatzprodukt Nagios Log Server erlaubt zentrale Analyse von Log-Dateien
- ✓ Wegen Kritik an Nagios Enterprises wurden mehrere Forks (alternative Versionen) wie Shinken, Icinga und Naemon ins Leben gerufen

5.1.5 Log-Analyse

Prozess	Instanz	Global
System	Echtzeit	Historisch
IT	Push	Pull
Datenbasiert	Nachrichtenbasiert	

In der Regel produzieren alle professionell eingesetzten IT-Systeme Log-Dateien. Werkzeuge zur Log-Analyse machen sich diesen kleinsten gemeinsamen Nenner zunutze, indem sie Systeme anhand der Log-Dateien überwachen und analysieren. Dabei kommen nicht nur Anwendungslogs von Webservern oder Skripten zum Einsatz, sondern auch Komponentenlogs (Firewalls, Datenbanken oder Betriebssystem-Logs)

Die Verwendung von Log-Dateien birgt allerdings etliche Herausforderungen. Log-Dateien enthalten eine hohe Anzahl an Informationen in zeitlich markierten Einträgen. Allerdings sind die einzelnen Log-Einträge zumeist unstrukturierte Meldungen. Auch ist ein Großteil der Meldungen für die Überwachung nicht relevant. Zudem werden Log-Dateien im Regelfall lokal auf Servern, Komponenten, etc. abgelegt und nicht an einer zentralen Stelle.

Log-Analyse-Werkzeuge aggregieren daher im ersten Schritt alle relevanten Log-Dateien eines verteilten IT-Systems (siehe Abbildung 27). Neue Log-Dateien werden periodisch abgerufen, ausgelesen und in ein einheitliches Format überführt. Alle Log-Einträge werden dann in einer gemeinsamen Datenbank gespeichert. Das Resultat ist ein einheitliches Log des Gesamtsystems.

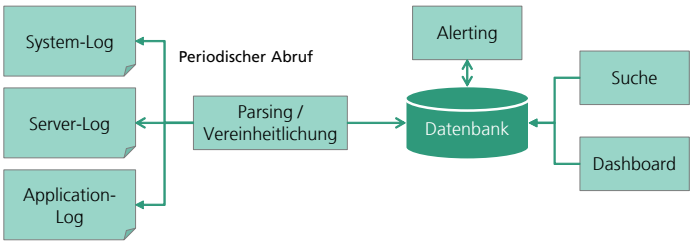

Abbildung 27: Typischer Aufbau eines Log-Analyse-Werkzeugs

Aufbauend auf diese Log-Datenbank bieten Log-Analyse-Werkzeuge weitere Funktionalität. Eine Alerting-Komponente kann bei bestimmten Log-Einträgen (oder bestimmten Mustern von Log-Einträgen) Alarme versenden. Eine Suchfunktion vereinfacht das Nachvollziehen von Abläufen im verteilten IT-System. Dashboards erlauben die Visualisierung von aus den Logs gewonnenen Daten und Kennzahlen. Kann man beispielsweise aus Log-Dateien den Beginn und das Ende eines Vorgangs ablesen, kann man Durchlaufzeiten, Erfolgsrate, etc. visualisieren.

	Inhalt
Vorteile	*+ Sehr umfangreiche Anbindung an verschiedenste Systeme* *+ Erstellung von Historien* *+ Visualisierung, Statistiken und Alarme* *+ Ansätze zur Ursachenerkennung durch umfangreiche Datenbasis* *+ Verbindet IT-, System- und Prozessebene* *+ Nachverfolgung über Systemgrenzen hinweg*
Nachteile	*- Keine Echtzeit* *- Vorhandensein und Zugriff auf Log-Dateien notwendig* *- Hoher Konfigurationsaufwand* *- Beschränkung auf Log-Inhalte*

Tabelle 14: Vor- und Nachteile von Log-Analysen

5.1.5.1 Technologie: ELK-Stack (Elasticsearch, Logstash, Kibana)

Der ELK-Stack ist eine Kombination aus drei Open-Source Produkten:

Name: ELK stack
Art: Log-Analyse
Firma: Elasticsearch (USA)
Lizenz: Open Source
Web: elastic.co/de/elk-stack

- ✓ **E**lasticsearch – Suche auf unstrukturierten Daten
- ✓ **L**ogstash – Verarbeitung von Log-Daten aus verschiedenen Quellen
- ✓ **K**ibana – Browser-Anwendung zur Visualisierung
- ✓ Sammeln von Log-Daten mittels sogenannter Beats (Agenten)
- ✓ Extraktion strukturierter Informationen (Timestamps, IPs, Prozess-Ids oder Fehlerstufe) mittels Werkzeug GROK
- ✓ Vereinheitlichung von Logs aus verschiedenen Quellen
- ✓ Suchen in verteilten Logs nach Stichworten, Freitext, Zeitintervallen oder Prozess-IDs
- ✓ Aggregation und Visualisierung von Log-Daten
- ✓ Erweiterbar mittels Plugin-Konzept

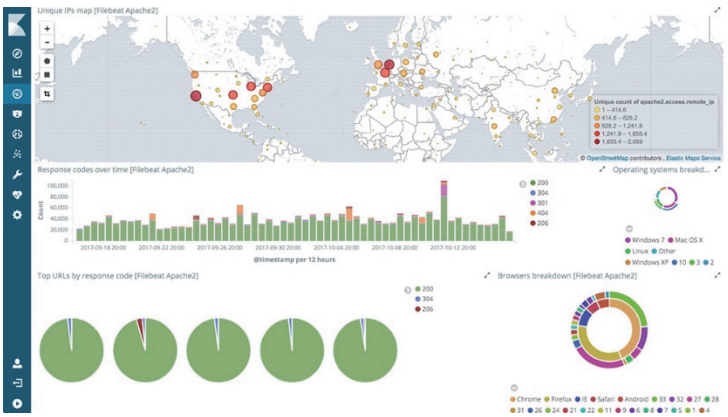

Abbildung 28: Auswertung von Server-Logs mittels ELK (Elasticsearch, 2019)

5.1.5.2 Technologie: Splunk

Splunk ist ein kommerzielles Werkzeug zur aggregierten Analyse von Log-Dateien in verteilten IT-Systemen.

Name: Splunk
Art: Log-Analyse
Firma: Splunk Inc. (USA)
Lizenz: Kommerziell
Web: splunk.com/de:de

- ✓ Auswertung mittels eigener Abfragesprache (Search Processing Language)
- ✓ Clustering und Korrelation von Log-Ereignissen verschiedener Quellen
- ✓ Visualisierung in Dashboards
- ✓ Alerting und Report-Erstellung
- ✓ Erweiterungen für IoT, Nutzerverhalten, IT-Sicherheit und Überwachung verteilter Geschäftsprozesse
- ✓ Automatische Erstellung von Historien
- ✓ Skalierbar dank Clustering
- ✓ Erweiterbar dank eigener Entwicklungsplattform
- ✓ Marktplatz für Erweiterungen (Apps)

5.1.5.3 Technologie: Zipkin

Zipkin ist ein Open-Source Werkzeug zum Erstellen von Traces in verteilten IT-Systemen. Traces sind Sequenzen von Aufrufen, die zu einer Anfrage gehören.

Name: Zipkin
Art: Tracing
Firma: Zipkin Project
Lizenz: Open Source
Web: zipkin.io

- ✓ Zipkin visualisiert Traces als Gantt-Diagramm Insbesondere für Microservice-Architekturen
- ✓ Datensammlung mittels Instrumentierung der Services mithilfe von Softwarebibliotheken (C, C#, Java, PHP, Ruby oder Python)
- ✓ Dadurch senden Services Tracing-Daten an Kollektoren
- ✓ Requests führen eine einmalige traceID mit, durch die eine Korrelation von Aufrufketten möglich ist

5.1.5.4 Technologie: Log-File-Viewer

Werkzeuge zur Betrachtung von Log-Dateien erlauben es, Log-Dateien komfortabler zu betrachten als in einem reinen Text-Editor. Obwohl diese Werkzeuge keine Überwachungs-Software im klassischen Sinne darstellen, seien Sie hier als vergleichsweise aufwandsarme Alternative genannt.

- ✓ Verbindung mehrerer Log-Dateien
- ✓ Filterung nach Zeit, Thread, Code-Datei, Klasse oder Paket
- ✓ Highlighting von Fehlern und Warnungen
- ✓ Zahlreiche Vertreter für verschiedene Log-Dateien
  - ➢ Apache Chainsaw (Java/Log4J) / MSBuild Log Viewer (Visual Studio)
  - ➢ Log Parser Studio (Microsoft Exchange) / PSI-Probe (Tomcat)
- ✓ Standalone-Werkzeuge
- ✓ Im Vergleich zu »vollwertigen« Log-Analyse Werkzeugen geringerer Installations- und Administrationsaufwand
- ✓ Kein Monitoring, Alerting, oder Ähnliches

5.1.6 Business Intelligence

Business Intelligence (BI) bezeichnet Methoden, Technologien und Software zur Gewinnung von Geschäftswissen (»Business Understanding«) aus Daten. Dieses Wissen dient unter anderem als Hilfe für das Treffen von Entscheidungen. (Richards et al., 2019) BI ist eine Unterkategorie der Fachbereiche Data Mining bzw. Data Science.

Prozess	Instanz	Global
System	Echtzeit	Historisch
IT	Push	Pull
Datenbasiert	Nachrichtenbasiert	

In Abbildung 29 ist der CRISP-DM Standardprozess für Data Mining aufgezeigt, der die Grundlage für viele Business-Intelligence Anwendungen ist. Zu Beginn steht eine Fragestellung, für die Geschäftswissen gewonnen werden soll, z. B. die Untersuchung des eigenen Geschäfts nach Dimensionen wie Quartalen, Regionen, Produktgruppen und Kundengruppen. Dabei sollen Korrelationen und kausale Zusammenhänge gefunden werden, z. B., um die häufigsten Gründe für das Scheitern eines Prozesses zu finden.

Um diese Fragestellungen zu beantworten, muss zunächst im Schritt »Data Understanding« ein Verständnis für die eigenen Daten gewonnen werden. Dabei werden potenzielle Datenquellen identifiziert und deren Präzision, Aktualität, Vollständigkeit, etc. untersucht. Historische Daten können aus Datenbanken, Dokumenten oder Log-Dateien gewonnen werden. Ist bekannt, wo die notwendigen Daten vorhanden sind (oder erfasst werden können), werden diese im Schritt »Data Preparation« für die Analyse vorbereitet. Da diese Datenquellen zumeist keine einheitliche Form aufweisen, müssen sie zunächst gesammelt und in ein gemeinsames Format konvertiert werden (Datenintegration).

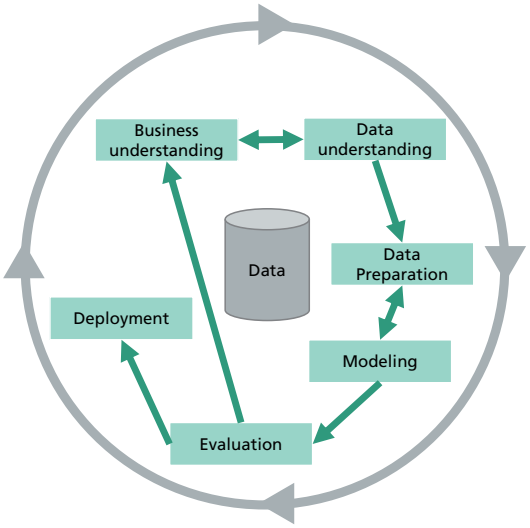

Abbildung 29: CRISP-DM Standardprozess für Data Mining

Abbildung 30 zeigt einen sogenannten ETL-Prozess zur Datenintegration, der in drei Schritten verläuft. Im Schritt »Extract« werden Daten aus ihren Quellen im Originalformat eingelesen. Im Schritt »Transform« werden die Daten umgewandelt und bereinigt. Dabei können z. B. redundante Datensätze entfernt, Datumsangaben in ein einheitliches Format übertragen werden oder auch unvollständige Datensätze aus sekundären Quellen ergänzt werden. Im Schritt »Load« werden die transformierten Daten in das Zielsystem geladen. Im Beispiel handelt es sich dabei um ein Data Warehouse, eine Datenbank, die auf Abfragen zu Analysezwecken optimiert ist.

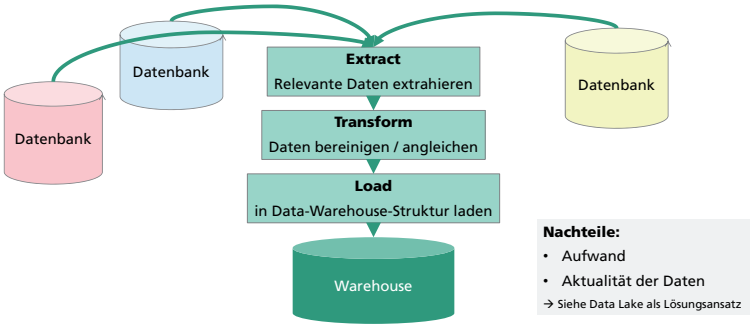

Abbildung 30: Extract-Transform-Load-Prozess zur Datenintegration

Die vereinheitlichten Daten werden in den Schritten »Modeling« und »Evaluation« ausgewertet. Je nach verwendeten Methoden des Data Minings unterscheiden sich diese Schritte stark. Wird z. B. Machine Learning verwendet, werden im Modellierungsschritt tatsächliche Modelle erstellt, um zum Beispiel den Ausgang neuer Geschäftsvorfälle vorauszusagen. In einfacheren Fällen werden in der Modellierung konventionelle statistische Methoden angewendet. Bei der Evaluation werden Daten und erstellte Modelle ausgewertet. Ergebnis der Evaluation ist neues Geschäftswissen, sowie in manchen Fällen weitere Artefakte wie z.B. prädiktive Modelle, die im »Deployment« in den Produktivbetrieb überführt werden.

Die Schritte des CRISP-DM Prozesses bilden einen Zyklus, der allerdings nicht strikt ist. Während der Durchführung ist es üblich, dass einzelne Schritte mehrfach durchgeführt werden. Auch Schritte »rückwärts« sind möglich. Bemerkt man beispielsweise, dass für die Modellierung noch Daten-Attribute fehlen, wird an die Datenintegration zurückgesprungen und erweitert diese entsprechend, sodass ein vollständiger Datensatz zur Verfügung steht. Business-Intelligence-Werkzeuge implementieren den CRISP-DM (in Teilen oder ganz) und unterstützen bei der Durchführung. Typische Features sind die grafische Modellierung von Workflows zur Datenintegration, Erstellen von Reports und Visualisierungen, sowie komplexe Auswertungen wie Prognosen und Ursachenanalyse.

	Inhalt
Vorteile	*+ Reiche Analyse-Möglichkeiten* *+ Historische Auswertungen und Prognosen* *+ Anbindung verschiedenster Datenquellen* *+ Möglichkeiten zur Root-Cause-Detection* *+ Verarbeitung großer Datenmengen* *+ Nachverfolgung über Systemgrenzen hinweg*
Nachteile	*- Keine Echtzeit* *- Hohe Datenqualität erforderlich* *- Für operative Störungserkennung ungeeignet*

Tabelle 15: Vor- und Nachteile von Business Intelligence

5.1.6.1 Technologie: Tableau

Tableau ist eine kommerzielle Business-Intelligence-Lösung, die eine Sammlung von Werkzeugen zur Verfügung stellt.

Name: Tableau
Art: Business Intelligence
Firma: Tableau Software (USA)
Lizenz: Kommerziell
Web: tableau.com

- ✓ Betrieb lokal oder in Cloud
- ✓ Komponenten für Datenzugriff
  - ➢ SQL, SAP, ODBC, JSON, Excel, etc.
- ✓ Interaktive Datenvorbereitung und Bereinigung
  - ➢ Grafische Definition eigener Workflows
- ✓ Umfassendes Berechtigungs- und Mandantenkonzept
- ✓ Viele Standardverfahren für Datenanalyse vorhanden
- ✓ Interaktive Datenvisualisierung, Aggregation und Analyse
- ✓ Individuelle Dashboards und Reports
- ✓ Unterstützung für diverse Cloud-Anbieter wie Amazon, Google, etc.
- ✓ Erweiterbar durch APIs

5.1.6.2 Technologie: Pentaho Platform

Die Pentaho Plattform (Hitachi, 2019) ist eine Business-Intelligence Software, die verschiedene Features anbietet: Datenintegration, OLAP Services, Data Mining oder Reporting. Folgende Eigenschaften können dieser Lösung zugeordnet werden:

Name: Pentaho
Art: Business Intelligence
Firma: Hitachi Vantara (USA)
Lizenz: Kommerziell
Web: pentaho.com

- ✓ Interaktive Datenvisualisierung
- ✓ Realtime-Datenverarbeitung
- ✓ Betrieb lokal oder in Cloud
- ✓ Bietet eine kostenlose Community und kostenpflichtige Enterprise Edition
- ✓ Pentaho Data Integration für Extract-Transform-Load
  - SQL, SAP, Big Data, JSON, Excel, etc.
  - Grafische Workflows zum Daten-Import
  - Einbindung von Machine Learning in den Daten-Import
  - Einbindung von Machine Learning in den Daten-Import
- ✓ Pentaho Business Analytics zur visuellen Datenauswertung
  - Große Anzahl an Visualisierungen
  - Erstellung individueller Dashboards und Reports
  - Optimiert auch für mobile Endgeräte
- ✓ Einbettung von Visualisierungen in externe Dokumente und Anwendungen

5.1.7 Business Activity/Process Monitoring Datenanbindungen

Prozess	Instanz	Global
System	Echtzeit	Historisch
IT	Push	Pull
Datenbasiert	Nachrichtenbasiert	

Als Business Activity Monitoring (BAM) oder Business Process Monitoring (BPM) wird die Überwachung von ausführbaren Geschäftsprozessen bezeichnet. Dabei liegt bei Business Activity Monitoring der Fokus auf einzelnen Aktivitäten und dazugehörigen Kennzahlen (»Key Performance Indikatoren« (KPIs)), während bei Business Process Monitoring die Grundlage der Betrachtung ein Geschäftsprozess ist, dessen Ablauf überwacht wird, wobei ebenfalls Kennwerte gemessen und Störungen erkannt werden.

Jedoch sind BAM bzw. BPM als Begriffe inzwischen etwas aus der Mode gekommen und daher weniger gebräuchlich, viele Produkte die zur Blütezeit des BAM auf den Markt kamen, sind inzwischen umbenannt worden oder werden nicht mehr gepflegt und sind somit Legacy-Anwendungen. Dennoch sollte man die diesen Technologien zugrundeliegenden Technologien nicht als veraltet abtun. Während frühe Lösungen zu BAM und BPM in der Praxis Schwierigkeiten mit dem mangelnden Reifegrad des Geschäftsprozessmanagements (engl. »Business Process Management«) in Unternehmen hatten, da beispielsweise Prozessmodelle nicht vorlagen oder mit dem realen Prozess nicht übereinstimmten, wissen aktuelle Versionen von BPM-Werkzeugen mit diesen suboptimalen real existierenden Rahmenbedingungen umzugehen.

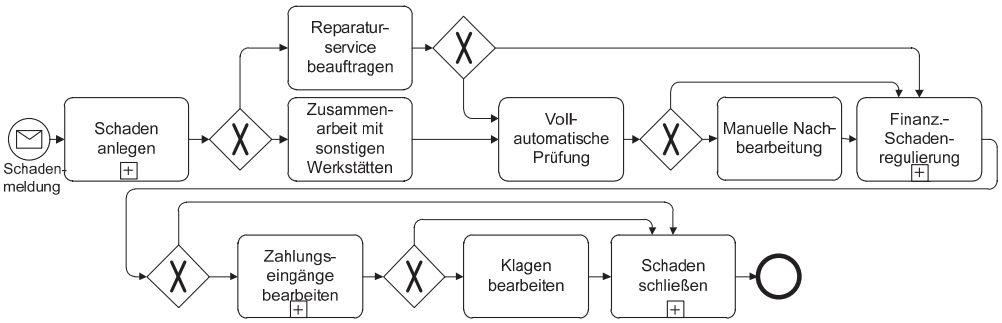

Abbildung 31: Beispielprozess aus der Versicherungsbranche, modelliert in BPMN

Bei BAM und BPM wird typischerweise ein aus mehreren Aktivitäten bestehender Geschäftsprozess überwacht, wobei der Fokus auf Geschäfts- und Prozesskennzahlen liegt. Dazu werden an den einzelnen Aktivitäten Messungen von Daten vorgenommen,

mittels denen der Ablauf einer Prozessinstanz nachverfolgt wird. Ein einfaches Beispiel ist, eine Messung zu Beginn und Ende eines Prozesses vorzunehmen. Mittels der ersten Messung (und des Ausbleibens der zweiten Messung), ist so feststellbar, dass ein Prozess innerhalb einer gewissen Frist nicht abgeschlossen wurde. Bei in einer Business Process Engine ausgeführten Prozessen können diese Messungen oft automatisch vorgenommen werden, da der Process Engine bekannt ist, welche real durchgeführte Aktivität mit welcher Aktivität im Prozessmodell korreliert.

Während unmittelbar nur Werte einzelner Prozessinstanzen (also der einzelnen Ausführung des Prozesses für einen Vorgang) gemessen werden, können gemessene Werte von Prozessinstanzen aggregiert werden, um Summen, Durchschnittswerte etc. zu bilden. So lässt sich ein Überblick über den Prozess als Ganzes ermitteln, beispielsweise die durchschnittliche Durchführungszeit einzelner Aktivitäten, die Erfolgsrate und das tägliche Prozessvolumen. Diese Daten können in Dashboards und Reports visualisiert werden. Werden im Prozess Abweichungen vom Soll-Zustand festgestellt, können auch Alarme erzeugt werden.

Die enge Kopplung zwischen Prozessausführung und Prozessüberwachung in einer Process Engine hat auch über eine reine Überwachung hinaus Nutzen. So ist es möglich, auf einen Fehlerfall zu reagieren. Wird zum Beispiel eine Maximalzeit überschritten, kann automatisch eine Aktivität erzeugt werden, die die überfällige Prozessinstanz wieder an einen Sachbearbeiter zuweist. Die technische Basis von BPM kann eine der bisher genannten Technologien sein. Möglich ist beispielsweise der Einsatz von Stream Analytics/Complex Event Processing. Diese zugrundeliegende Technologie ist dann aber »gekapselt«, sodass der Endanwender auf Prozessebene arbeiten kann.

	Inhalt
Vorteile	*+ Prozess-Fokus bei der Überwachung* *+ Direkte Anbindung von Business Process Engines* *+ Verbindung zwischen Messdaten und Prozess für Geschäftsanwender* *+ Behandlung/Kompensation von Fehlerfällen möglich*
Nachteile	*- Bei vielen Anbietern inzwischen Legacy-Technologie* *- Anwendung muss korrekt modellierter Prozess sein oder als Prozess modelliert werden* *- Hoher manueller Einrichtungsaufwand* *- Kein Fokus auf Fehlerfälle*

Tabelle 16: Vor- und Nachteile des Business Process/Activity Monitoring

5.1.7.1 Camunda BPM

Camunda BPM ist eine moderne Plattform für Geschäftsprozessmanagement. Folgende Eigenschaften können dieser Lösung zugeschrieben werden:

Name: Camunda BPM
Art: Business Process Management
Firma: Camunda Services GmbH (DE)
Lizenz: Kommerziell / Open Source
Web: camunda.com

- ✓ Komponenten für die Modellierung und Ausführung von Geschäftsprozessen
- ✓ Business Process Model and Notation (BPMN)
- ✓ Cockpit als Werkzeug für Business Process Monitoring
- ✓ Analyse von Prozessinstanzen, Fehlererkennung/-korrektur
- ✓ Dashboard zur Anzeige von begrenzten Prozesskennzahlen
- ✓ Open Source oder kommerzielle Lizenz
- ✓ Schnittstellen über REST oder JAVA
- ✓ Unterstützung bei der manuellen Prozessdurchführung

5.1.7.2 Oracle Business Activity Monitoring

Oracle Business Activity Monitoring ist eine Überwachungslösung für Geschäftsprozesse. Folgende Eigenschaften sind dieser Lösung zuzuschreiben:

Name: Oracle BAM
Art: Business Activity Monitoring
Firma: Oracle (USA)
Lizenz: Kommerziell
Web: oracle.com/technetwork /middleware /bam/overview/index.html

- ✓ Anzeige von historischen Daten, z. B. aus Datenbanken
- ✓ Anzeige von Echtzeitdaten, sofern Anwendungssysteme diese übermitteln
- ✓ Anbindung an existierendes Business Process Management (BPM) oder serviceorientierte Architekturen (SOA) von Oracle
- ✓ Berechnung von Key Performance Indikatoren
- ✓ Vorgefertigte Dashboards für Standard Use Cases
- ✓ Grafisch konfigurierbare individuelle Dashboards
- ✓ Drilldown und Analyse historischer Daten
- ✓ Wird seit 2014 nicht mehr aktiv weiterentwickelt

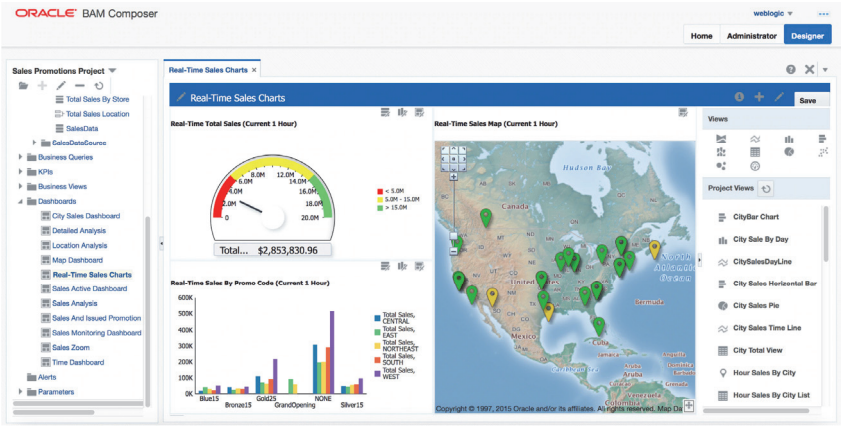

Abbildung 32: Übersicht über das BAM Composer von Oracle (Oracle, 2019a) [5]

5.1.8 Application Performance Management

Werkzeuge zum Application Performance Management (APM) dienen dazu, die fachliche Funktionalität einer Anwendung unter allen Gesichtspunkten zu überwachen. Diese Kategorie von Werkzeugen ist im Vergleich zu den anderen beschriebenen relativ neu und ist speziell für hochskalierbare, verteilte IT-Systeme entwickelt worden. APM-Werkzeuge sollen also eine Antwort auf die Komplexitätslücke zwischen konventionellen Überwachungswerkzeugen und Cloud-Anwendungen sein.

APM-Werkzeuge haben dabei den Anspruch, alle Aspekte einer Anwendung von der physischen Hardware bis hin zur User Experience abzudecken. Dieser allumfassende Ansatz wird im Englischen auch als »Full Stack« bezeichnet. Dazu bedienen sich APM-Werkzeuge Kombinationen von Technologien wie Stream Processing, IT-Infrastruktur-Überwachung und Log-Analyse. Beispiele sind die automatische Erkennung der Komponenten des verteilten IT-Systems und das automatische Mapping von Anwendungen auf die IT-Infrastruktur, wobei Beziehungen und Datenflüsse zwischen einzelnen Komponenten ebenfalls erkannt werden sollen. So entsteht eine »Landkarte« des verteilten IT-Systems, ohne dass der Nutzer Komponenten eigenhändig einpflegen muss.

Anwendungen werden auf verschiedene Weisen überwacht. So werden beispielsweise konventionelle Agenten eingesetzt, aber auch komplexere Überwachungsmechanismen

[5] Copyright Oracle and its affiliates. Used with permission.

wie der »Drill-Down« in eine Anwendungsumgebung, mit der der Programmfluss der Anwendung bis hinunter auf Quelltextebene nachverfolgt wird. Über die Performance einer Anwendung hinaus wird auch die User Experience überwacht. Mittels Tracking-Skripten auf Webseiten kann beispielsweise die Customer Journey eines einzelnen Kunden überwacht werden. Meldet dieser sich beim Support, kann Schritt für Schritt nachvollzogen werden, was der Kunde getan hat und wo das Problem auftrat. Diese Werte können auch aggregiert werden. Zugriffszahlen und Aufrufzeiten werden so z. B. geografisch geclustert, sodass festgestellt werden kann, dass die eigene Webseite gerade aus Osteuropa nicht erreichbar ist oder lange Ladezeiten hat. APM-Werkzeuge bringen auch Features zur Ursachenerkennung in verteilten IT-Systemen mit. So sollen beispielsweise überlastete Komponenten (»Bottlenecks«) erkannt werden. Schließlich werden auch geschäftliche Kennzahlen und Key Performance Indikatoren erfasst.

	Inhalt
Vorteile	*+ Betrachtung des Kundenerlebnisses* *+ Überwachung von Anwendungen und IT-Infrastruktur* *+ Root Cause Detection unter Betrachtung der Zusammenhänge* *+ (Begrenzte) Überwachung von Dritten möglich* *+ Unterstützt viele Standard-Technologien*
Nachteile	*- Ausgelegt auf Web- und Cloud-Anwendungen* *- Fokus auf Performance mit Einschränkungen verbunden* *- Verwendung von Agenten und JavaScript auf Webseiten notwendig*

Tabelle 17: Vor- und Nachteile des Application Performance Managements

5.1.8.1 Dynatrace

Dynatrace ist ein kommerzielles Application Performance Management-Werkzeug, das sowohl als Cloud-Lösung als auch als lokale Installation (»On-Premise«) verfügbar ist. Folgende Eigenschaften sind dieser Lösung zuzuschreiben:

Name: Dynatrace
Art: Application Performance Management
Firma: Dynatrace (US)
Lizenz: Kommerziell
Web: dynatrace.com

- ✓ Integration von Systemen und Web-Anwendungen über Agenten
- ✓ Unterstützung für verschiedene Betriebssysteme, Webserver, Datenbanken und Cloud-Umgebungen
- ✓ Automatisches Mapping von Anwendungen auf IT-Infrastruktur mit Visualisierung der Beziehungen und Datenflüsse
- ✓ Überwachung des Nutzerverhaltens und Erlebens (z. B. Verweildauer auf Webseite, Wartezeiten) mittels JavaScript
- ✓ Festlegen von automatischen Überprüfungen aus Nutzersicht, zum Beispiel Durchführung einer Buchung im Webfrontend
- ✓ Deep Dive – Detailliertes, internes Monitoring von Anwendungen
- ✓ Replay des Kundenerlebnisses für einzelne Kunden
- ✓ Überwachung von Dienstleistern und Partnern
- ✓ Root Cause Detection bei Nutzerproblemen

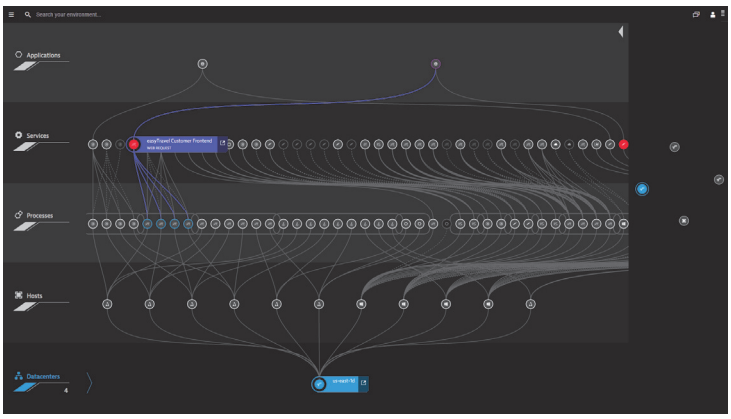

Abbildung 33: Fehlerlokalisierung in Dynatrace (Dynatrace, 2019)

5.1.8.2 AppDynamics

AppDynamics ist ein kommerzielles Application Performance Management-Werkzeug. Folgende Eigenschaften sind dieser Lösung zuzuschreiben:

Name: AppDynamics
Art: Application Performance Management
Firma: AppDynamics LLC (US)
Lizenz: Kommerziell
Web: appdynamics.com

- ✓ Integration von Systemen und Web-Anwendungen über Agenten
- ✓ Unterstützung für verschiedene Betriebssysteme, Webserver, Datenbanken und Cloud-Umgebungen
- ✓ Automatisches Mapping von Anwendungen auf IT-Infrastruktur mit Visualisierung der Beziehungen und Datenflüsse
- ✓ Überwachung des Nutzererlebens mittels JavaScript im Browser oder in mobiler App
- ✓ Festlegen von Überprüfungen aus Nutzersicht mittels automatisiertem Web-Browser
- ✓ Erkennen und Visualisieren von anormalen Abläufen sowie Root Cause Detection
- ✓ Internes Monitoring von Anwendungen (z. B. auf Klassenbasis in Java)
- ✓ Offener Marktplatz für Plug-Ins

5.1.9 Zusammenfassung

Der Überblick über aktuelle Anwendungen und Technologien zeigt, dass zahlreiche Lösungen für Störungserkennung und Überwachung auch in verteilten IT-Systemen existieren, die einen hohen Reifegrad aufweisen. Zwar kann keine Softwarelösung alle Fragestellungen und Anwendungsszenarien abdecken. Aber viele Softwarelösungen sind erweiterbar und kombinierbar.

Die Auswahl von Lösungen hängt stark von den zu beantwortenden Fragestellungen bei der Überwachung ab, weswegen eine Auswahl immer anhand individueller Ziele erfolgen muss. Je nach Unternehmen ist es auch möglich, dass verschiedene Zielgruppen unterschiedliche Werkzeuge brauchen: Kunde, Manager, First Level/Second Level Support, Entwickler, Retail, etc. Ältere Lösungen wie SNMP und Complex Event Processing haben inzwischen einen hohen Reifegrad und eine breite Unterstützung am Markt, erfordern aber in komplexeren Cloud-Umgebungen hohe Integrations- und Konfigurationsaufwände. Moderne Lösungen integrieren diese bewährten Technologien oft als Basistechnologie, z. B. SNMP in PRTG und Nagios oder Complex Event Processing in TIBCO Business Events.

Business Intelligence bietet umfangreichste Analysen, ist für operative Störungserkennung aber nicht geeignet. Reine IT-Infrastruktur-Überwachung weist einen hohen Grad an Reife und Standardunterstützung auf und kann im Vergleich zu anderen Werkzeugen mit geringem Aufwand eingeführt werden. Moderne Log-Tracing-Werkzeuge können die Untersuchung von Log-Dateien in verteilten IT-Systemen stark vereinfachen und darüber hinaus ohne weitere Integration zusätzliche Informationen gewinnen. Application Performance Management ist eine vergleichsweise neue Kategorie Werkzeug, die spezifisch auf die wachsende Komplexität von unternehmensübergreifenden Cloud-Umgebungen ausgelegt ist und verbindet verschiedene Technologien wie IT-Infrastruktur-Überwachung und Log-Tracing. Im Gegenzug ist bei Business Process/Activity Monitoring-Werkzeugen Vorsicht angebracht. Manche Werkzeuge werden nicht mehr weiterentwickelt und sind für neue Projekte nicht zu empfehlen, sodass man prüfen sollte, inwieweit der Hersteller diese noch mit neuen Features und Versionen unterstützt.

Mit der am Markt existierenden Vielzahl von Lösungen ist eine Individualentwicklung von Softwarelösungen für die Störungserkennung und Überwachung nur noch schwer zu begründen. Es ist unwahrscheinlich, dass man mit vertretbarem Aufwand einen Vorteil gegenüber dem Einsatz einer existierenden Lösung hat. Für Standard-Probleme sollte daher Standardsoftware eingesetzt werden, während für komplexe, anwendungsfallspezifische Probleme die Entwicklung individueller Lösungs-Kombinationen notwendig sein kann, wobei auf Standard-Technologien aufgesetzt werden sollte.

5.2 Standards

Mit Blick auf verteilte IT-Systeme und ihre einzelnen zusammenspielenden Komponenten spielen Standards für einen reibungslosen Betrieb eine Schlüsselrolle. Dazu gehören Standards in den Protokollen, den Schnittstellen und der Datenhaltung. Jedoch zeigt die Praxis, dass die Einhaltung und die Verwendung von Standards sich als schwierig darstellen. Dies liegt darin begründet, dass sich verteilte IT-Systeme aus zahlreichen unterschiedlichen Software- und Hardware-Komponenten zusammensetzen, die mit unterschiedlichen oder sogar auch mit veralteten und unvollständigen Standards arbeiten. Statt auf Standardisierung setzen die meisten Werkzeughersteller auf Interoperabilität und binden über Plug-Ins o.Ä. zahlreiche Standards an und lassen sich um weitere Standards erweitern.

Im Folgenden wird nun auf einige Standards eingegangen, die in verteilten IT-Systemen eine Rolle spielen. Dazu gehören Fehlercodes und das OSI-Modell.

5.2.1 Fehlercodes

In der IT existieren zahlreiche standardisierte Fehlercodes. Diese können Protokoll-basiert sein (z. B. HTTP, FTP, TCP oder ICMP) oder auch Produkt-basiert (z. B. SQL). Für viele Anwendungen gibt es keine standardisierten Fehlercodes für Individuelle Fehlermeldungen oder für Fehlerprotokollierung in Log-Dateien. Insbesondere gibt es keine übergreifenden Standards für Fehlercodes. Fehlercodes sind also immer auf eine einzelne Komponente, Protokoll oder Software spezifisch. Allerdings lassen sich mit Netzwerkprotokoll-Fehlern viele Szenarien in verteilten IT-Systemen abdecken.

Der Vorteil bei der Verwendung von Fehlercodes ist es, dass Fehler klarer zwischen verschiedenen Parteien kommuniziert werden können. Ebenso wird durch die Vergabe von Fehlernummern während der Entwicklung sichergestellt, dass alle vorhergesehenen Fehler des Systems auch dokumentiert werden. Während es keinen universellen Standard für Fehlercodes gibt, kann es sinnvoll sein, einen firmeninternen Standard zu entwickeln.

5.2.2 OSI-Modell

Das Open System Interconnection (OSI) Modell ist die Grundlage für offene Kommunikationsstandards. In Abbildung 34 ist der Aufbau des OSI-Modells aufgezeigt. Dieses hat sieben Abstraktionsschichten. Zu jeder Schicht gehören bestimmte Protokolle und Funktionen. Kommunizieren zwei Systeme miteinander, durchläuft die Kommunikation zweimal alle Schichten über die zwei Systeme hinweg. (Elektronik Kompendium, 2019). Die verschiedenen Abstraktionsschichten trennen unterschiedliche Aufgaben voneinander, sodass beispielsweise die höheren Schichten, mit denen ein Computer im Netzwerk kommuniziert gleichbleiben können, egal, ob auf den unteren Schichten eine drahtlose oder kabelgebundene Netzwerktechnologie eingesetzt wird.

Moderne verteilte IT-Systeme basieren auf dem OSI-Modell, wobei allerdings mehrere Schichten zusammengefasst werden, wie es beispielsweise bei TCP/IP der Fall ist.

Das Modell ist für die Störungserkennung insofern relevant, als dass auf jeder Schicht unterschiedliche Fehler auftreten können, die gesondert betrachtet werden müssen. Zum Beispiel kann der Aufruf einer Webseite scheitern, weil die physische Netzwerkleitung beschädigt ist, weil der Server abgestürzt ist, die Anfrage nicht rechtzeitig beantwortet werden kann oder kein gültiges SSL-Zertifikat vorhanden ist. Das OSI-Modell stellt insofern eine gute Planungs-/Diagnosegrundlage für Netzwerkverbindungen dar.

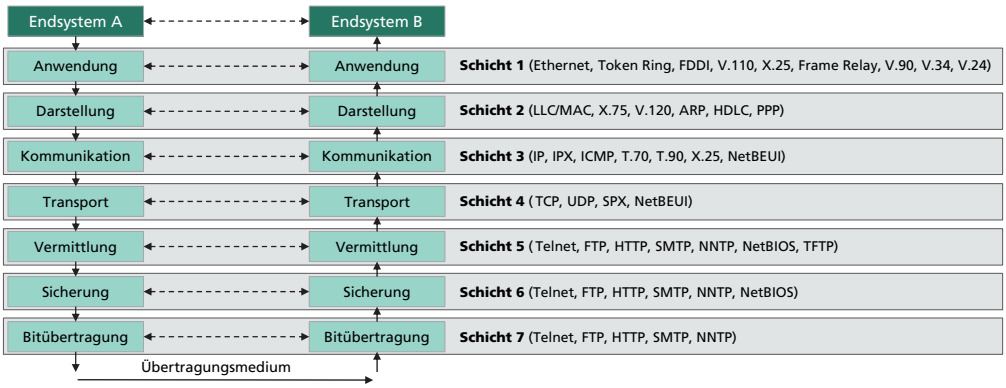

Abbildung 34: Aufbau des OSI-Modells (Elektronik Kompendium, 2019)

5.3 Software-Qualitätssicherung

Qualitätssicherung von Softwaresystemen und Störungserkennung sind miteinander verbunden. Zum einen sind Störungen auch durch Fehler in der Software bedingt, die zum Teil im Laufe der Entwicklung gemacht wurden. Zum anderen stellen Entwurfs- und Entwicklungsentscheidungen frühzeitig Weichen für die Störungserkennung, beispielsweise dafür, welche Möglichkeiten zum Monitoring bestehen.

Methoden zur Software-Qualitätssicherung lassen sich in drei Dimensionen gliedern (Frühauf et al., 2002):

- **Organisatorisch**
  - Projektmanagement in Entwicklungsprojekten
  - Organisation von Entwicklung, Betrieb und Wartung
  - Prozesse und Vorgehensmodelle
- **Konstruktiv**
  - Methoden des Software-Engineering
  - Konstruktion testbarer Software
  - Software-Qualität
- **Analytisch**
  - Code-Reviews
  - Software-Tests
  - Monitoring

Im Folgenden werden grundlegende, für die Störungserkennung relevante Aspekte sowie aktuelle Methoden kurz beschrieben.

5.3.1 Software-Qualität nach ISO 25010

ISO 25010 ist eine Norm für funktionale und nicht-funktionale Qualitätsmerkmale von Software (ISO/IEC, 2010). Viele dieser Merkmale haben Auswirkungen auf den Betrieb. Merkmale sind in der Norm allerdings rein qualitativ beschrieben. Zur Quantifizierung von Merkmalen benötigt man Metriken.

In Abbildung 35 sind die in der Norm beschriebenen Qualitätsmerkmale aufgeführt. Für die Störungserkennung besonders relevante Merkmale sind fett gedruckt. Besonders hervorzuheben sind die Merkmale Analysierbarkeit (engl. Analysability) und Testbarkeit (engl. Testability). Diese sind in der Norm unter Wartbarkeit aufgeführt, haben allerdings auch eine Auswirkung auf den Betrieb.

Abbildung 35: Qualitätsmerkmale nach ISO 25010

Um Software im Betrieb zu überwachen, ist es notwendig, Informationen über den Zustand der Software zu gewinnen. Weiterhin kann es förderlich sein, zur Laufzeit die Funktionsweise durch Tests zu validieren (z. B. eine Beispielanfrage). Damit dies möglich ist, muss die Software entsprechende »Features« aufweisen. Diese beiden Qualitätsmerkmale werden im Folgenden auch unter dem Oberbegriff Beobachtbarkeit zusammengefasst.

5.3.2 Beobachtbarkeit

Die Beobachtbarkeit (engl. Observability) ist ein Begriff aus der Kontrolltheorie. Observability ist das Maß, inwieweit sich der Zustand eines Systems aus dessen Inputs und Outputs ableiten lässt (Gopal, 1993). Abbildung 36 zeigt den Zusammenhang zwischen Zustand, Input, Output und Beobachtbarkeit.

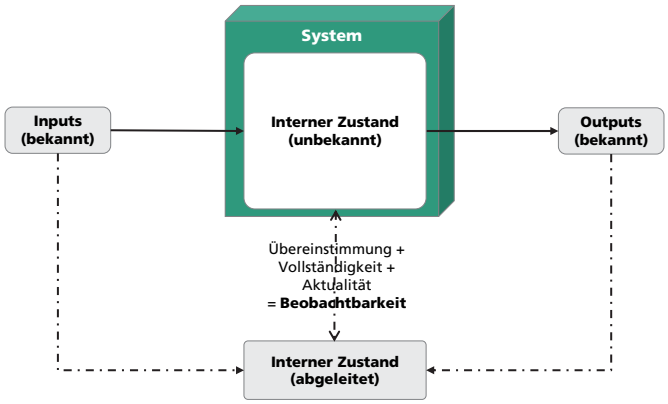

Abbildung 36: Definition der Beobachtbarkeit eines Systems

Jedes Softwaresystem ist im Betrieb zunächst eine sogenannte »Black Box«, deren interne Arbeitsweise nicht bekannt ist. Nur in der Interaktion mit dem System zeigt sich dessen Zustand. In der Praxis hat jedes Softwaresystem Schnittstellen zum Input und Output, zumindest diejenigen, die notwendig sind, damit es seine Funktion erbringt. Bei einer Windows-Anwendung ist dies zum Beispiel die grafische Benutzungsoberfläche. Bei einem Webservice sind es die Methoden, die dieser Webservice anbietet. Neben diesen funktionalen Schnittstellen gibt es auch noch weitere Inputs und Outputs eines Systems, die für die Beobachtbarkeit eine Rolle spielen.

Beispiele sind Log-Dateien, Datenbanken, versendete Nachrichten, geschriebene Dateien, Administrationswerkzeuge, Arbeitsspeicher und Ausführungsumgebung.

Jede Aussage über den Zustand des Systems muss auf den vorhandenen Outputs beruhen. Dies gilt auch für die Überwachung und Störungserkennung. Verschiedene Outputs unterscheiden sich allerdings voneinander in Bezug auf die in Tabelle 18 beschriebenen Qualitäten.

Art der Qualität	Beschreibung
Aktualität	*Wie aktuell sind die angegebenen Zustandsinformationen? Beispiel: Log-Datei wird nur alle 60 Sekunden geschrieben.*
Korrektheit	*Stimmen die Ausgaben mit dem Zustand überein? Beispiel: Unterschiedliche Werte in Datenbank und Arbeitsspeicher.*
Vollständigkeit	*Sind alle Daten in ausreichender Genauigkeit vorhanden? Beispiel: Ausführender Benutzer eines Vorgangs wird nicht gespeichert.*
Verfügbarkeit	*Wann, wo und wie lange ist ein Zugriff möglich? Beispiel: Lokale Log-Datei auf Server wird 24 Stunden aufbewahrt.*
Strukturiertheit	*In welcher Datenstruktur sind die Daten vorhanden? Beispiel: Anfrage liegt als Datensatz in SQL oder nur als PDF vor.*

Tabelle 18: Qualitäten einer Datenquelle

Was bedeutet dies in der Praxis? Für die Überwachung und Störungserkennung können nur diejenigen Daten verwendet werden, die verfügbar sind. Sind diese Daten unvollständig oder inkorrekt, bleiben Störungen unbemerkt oder ihre Ursache kann im Nachhinein nicht mehr ermittelt werden.

Beobachtbarkeit sollte daher schon bei der Implementierung von Software berücksichtigt werden, sodass alle notwendigen Daten aktuell, korrekt, vollständig und in einfach zu verarbeitenden Datenstrukturen verfügbar sind. Für bereits bestehende Systeme hilft dieser Rat natürlich nichts. Sogenannte »Legacy-Systeme« besitzen oft keine einfach zugänglichen Schnittstellen zur Messung. Um diese »Datensilos« zu öffnen, muss auf bestehende Outputs zurückgegriffen werden. Mögliche Techniken dafür sind in Tabelle 19 aufgelistet.

Technik	Beschreibung
Log-Tracing	*(Inkrementelles) Einlesen der Log-Dateien, während sie geschrieben werden, um wieder strukturierte Daten zu gewinnen.*
Datenbank-abfragen	*Wiederkehrende Abfragen auf Datenbanken, die das Software-System verwendet.*
Dateien	*Beobachtung von Ausgabeordner und Ausgabedateien auf Festplatte, Mitführen von Änderungen und Änderungszeitpunkten.*
Scraping	*Gewinnung strukturierter Informationen aus Benutzeroberflächen.*
Scripting	*Automatische Bedienung grafischer oder textbasierter Benutzerschnittstellen, um Informationen zu gewinnen.*
Daten-Request	*Anfrage an bestehende Schnittstelle zur direkten Datenabfrage.*

Dummy-Request	*Durchführung eines Prozesses an Schnittstelle zur Informationsgewinnung, z. B. Messung der Durchlaufzeit.*
Proxy	*Abhören / Mitprotokollieren ein- und ausgehender Netzwerknachrichten.*
Diagnose-funktionen	*Direkter Zugriff auf die Ausführungsumgebung (Java Virtual Machine, Webserver oder Betriebssystem).*
Tracking	*Einbinden von JavaScript, Zählpixel o.Ä. in Web-Frontend, das bei jedem Aufruf ebenfalls aufgerufen wird.*

Tabelle 19: Techniken zur Beobachtbarkeit von Systemen

Falls notwendige Daten nicht über bestehende Outputs verfügbar sind, muss die Anwendung erweitert werden. Allgemein ist abzuwägen, ob eine komplexe Integration bestehender Outputs oder eine Schaffung neuer Outputs die effizientere Alternative ist. Moderne Lösungen zur Überwachung und Störungserkennung bringen eine Vielzahl von Adaptern mit, die es erlauben, bestehende Outputs auszulesen und auszuwerten.

5.3.3 Metriken für Software-Qualität

Metriken dienen dazu, Software-Qualitätsmerkmale zu quantifizieren. In der Literatur gibt es eine Vielzahl solcher Metriken, eine Übersicht liefert beispielsweise (Koetter et al., 2019). Manche Metriken wie z. B. Anzahl der Codezeilen LOC (Lines of Code) lassen sich direkt berechnen. Aber auch komplexere Metriken existieren, z. B. bewertet LCOM (Lack of Cohesion in Methods) die Aufteilung von Programmcodes in einzelne Klassen. Metriken dienen der statischen Software-Analyse, das heißt, sie bewerten den Code, ohne ihn auszuführen. Dabei bewerten Metriken die Struktur des Codes, nicht jedoch dessen fachliche Korrektheit. Verbunden mit Metriken sind sogenannte »code smells« Stellen, die auf stilistische Fehler, Bugs oder Schwachstellen im Code hindeuten. Beispiele dafür sind nicht geloggte Fehler, große Mengen auskommentierter Codes oder tief verschachtelte Codes.

Es existieren zahlreiche, fertig verwendbare Kataloge von Metriken, die ohne großen eigenen Aufwand eingesetzt werden können sowie Softwarewerkzeuge zur statischen Analyse, die Metriken berechnen und Code-Stellen mit »smells« identifizieren. Verbreitete Werkzeuge sind:

- **SonarQube:** Open Source, Fokus auf Finden von Programmierfehlern und Schwachstellen
- **SourceMeter:** Kommerziell, umfangreiche Metrikberechnung
- **FindBugs:** Open Source, findet/priorisiert Fehlermuster in Java-Programmen

5.3.4 Testbarkeit von Software und Software-Tests

Was die Überwachung im Produktivbetrieb ist, ist der Test in der Software-Entwicklung. Tests überprüfen eine Software auf eine korrekte Arbeitsweise, das heißt, auf die Funktionalität gemäß der Spezifikation. Ziel eines Tests ist es, Software-Fehler zu aufzudecken. Die »Badewannenkurve« beschreibt ein Prinzip der Softwareentwicklung. Das besagt, je früher ein Fehler in der Softwareentwicklung gemacht wird, desto später im Prozess wird dieser entdeckt. Ein Syntaxfehler fällt dem Programmierer bereits beim nächsten Kompiliervorgang auf, während eine Fehlannahme bei der Spezifikation vielleicht erst im Betrieb auffällt. Deswegen sollten in jedem Schritt des Software-Entwicklungsprozesses qualitätssichernde Maßnahmen durchgeführt werden.

Tests können dadurch erleichtert werden, dass Software von vornherein so geschrieben wird, dass sie einfach testbar ist. Ansätze wie Test-Driven Development gehen sogar so weit, dass der Test vor der eigentlichen Software entwickelt wird. In Tabelle 20 werden kurz aktuelle Best Practices und Ansätze im Bereich von Softwaretests beschrieben.

Ansatz	Beschreibung
Clean Code	*Entwurfsmuster und Programmierprinzipien für wartbaren, testbaren und erweiterbaren Code (Martin et al., 2009).*
Test-Driven Development	*Implementierung von Tests (auch automatisiert) vor funktionalem Code in kleinen Iterationen (Janzen and Saiedian, 2005). So kann eine große Testabdeckung erreichen werden.*
Defensive Programming	*Paradigma, um sicherzustellen, dass Code unter allen Umständen funktioniert, indem möglichst viele Vorbedingungen und Eingaben auf potenzielle Fehler überprüft werden. (McConnell, 2009)*
Unit Testing	*Zerlegen von Programmcode in kleine, isoliert testbare Einheiten, die mittels sogenannter Unit Tests automatisiert getestet werden können. (Osherove, 2014)*
Regression Testing	*Testen bestehender Funktionalität in neuen Softwareversionen, um sicherzustellen, dass diese weiterhin korrekt funktioniert. (Desikan and Ramesh, 2006)*
S.O.L.I.D.	*Prinzipien für das Design objekt-orientierter Software. (Blokdyk, 2018)*
Developer Testing	*Wie kann bei der Entwicklung Rücksicht auf Tests genommen und testbarer Code geschrieben werden? (Tarlinder, 2017)*
Feature Flagging	*Neue Features werden so implementiert, dass sie mit einer Bedingung aktivierbar und deaktivierbar sind. Dies erhöht die Sicherheit beim Versionswechsel und ermöglicht A/B-Testing und Canary Testing. (Echagüe and Aijaz, 2018)*
A/B-Testing	*Vergleich von zwei Alternativen, indem Nutzer willkürlich Alternative A oder B verwenden, danach Vergleich (z. B. Performance/Zufriedenheit). (Siroker and Koomen, 2013)*
Canary Testing	*Ein Spezialfall des A/B-Tests. Eine neue Softwareversion wird einer zufällig gewählten kleinen Auswahl von Nutzern live geschaltet – diese dienen als unwissentliche Tester vor dem eigentlichen Versionswechsel. (Rajesh, 2017)*
Exception Tracking	*Sammeln von Fehlern einzelner Anwendungen an einem zentralen Punkt – Fehler werden gruppiert, sind filterbar und werden einzelnen Softwareversionen zugeordnet. (Setter, 2017)*

Tabelle 20: Übersicht Best Practices und Ansätze im Bereich von Softwaretests

6 Stand der Praxis

Die Untersuchung des aktuellen Stands in Forschung und Technik hat gezeigt, dass eine Vielzahl reifer Anwendungslösungen und Technologien existieren, um Störungen in verteilten IT-Systemen zu erkennen. Darauf aufbauend stellt sich zum einen die Frage, inwieweit diese Lösungen in der Praxis angekommen sind und zum anderen, welche Erfolge und Herausforderungen es dabei gegeben hat. Um ein möglichst umfassendes Bild des Stands in der Praxis zu gewinnen, wurden Experteninterviews mit verschiedenen Stakeholdern durchgeführt. Im Folgenden werden die Ergebnisse dieser Experteninterviews zusammengefasst.

6.1 Methodik

Für die Untersuchung des Stands der Praxis wurde als Methode das Experteninterview gewählt. Experteninterviews können als Orientierungshilfe in einem neuen Forschungsgebiet eingesetzt werden, um anhand des Expertenwissens Theorien über das Forschungsgebiet zu gewinnen. (Bogner and Menz, 2002)

Als Grundlage für die Experteninterviews diente ein Leitfaden, der anhand des aktuellen Stands von Forschung und Technik entwickelt wurde. Ein Leitfaden erlaubt im Vergleich zu einem Fragebogen eine offenere Befragung, die zwar eine systematische Vorgabe für die Interviewdurchführung gibt, aber ein freies Gespräch ermöglicht. Daher werden solche Interviews auch als semi-strukturierte Interviews bezeichnet. Der Grundgedanke des Leitfadens ist »So offen wie möglich, so strukturierend wie nötig« (Helfferich, 2019). Je nach Interviewpartner können der Ablauf und der Schwerpunkt des Gesprächs so dynamisch angepasst werden, um das Spezialwissen des Experten optimal erfassen zu können (Lethbridge et al., 2005). Der Nachteil solcher semi-strukturierter Interviews ist jedoch ein erschwertes Ermitteln von quantitative Aussagen. Für die Ableitung von Handlungsempfehlungen sind diese allerdings nicht notwendig.

Die Interviews wurden über einen Zeitraum von drei Monaten durchgeführt: Falls möglich persönlich, ansonsten telefonisch. Als Interviewdauer wurden 30 Minuten angesetzt, allerdings dauerte das Gespräch im Schnitt länger. Bei jedem Interview waren mindestens zwei Interviewer beteiligt, damit eine Protokollierung von relevanten Informationen sichergestellt wurde. Zur Auswahl der Experten wurden schon während der state-of-the-art-Analyse relevante Unternehmen identifiziert, die als Anbieter, Anwender oder Berater am deutschen Markt tätig sind. Darüber hinaus wurden auf einer internationalen Konferenz auch Experten außerhalb von Deutschland interviewt.

Der Inhalt des Gesprächsleitfadens orientiert sich an den folgenden Schwerpunkten, die jedoch nicht von jedem Interviewpartner gleich priorisiert beantwortet wurden:

- Spezifikation
- Service-Level
- Internet of Things (IoT)
- Datenintegration
- Überwachung von Systemen Dritter
- Störungserkennung
- Standardisierung
- Anpassung und Erweiterung
- Entwicklungsprozess
- Betriebsprozess
- Herausforderungen
- Zukünftige Entwicklungen

Für jedes Experteninterview wurde ein Protokoll erstellt und anonymisiert.

Die Auswertung der Protokolle fand mit Hilfe der qualitativen Inhaltsanalyse statt (Loosen, 2016, pp. 144–145). Dazu wurden Aussagen aus den Interviews extrahiert und verschiedenen Kategorien zugeordnet, die wiederum zu Schwerpunkten aggregiert wurden. Weiterhin wurde nach Übereinstimmungen und Widersprüchen zwischen den Experten gesucht, um Aussagen dementsprechend zu gewichten. Ergebnisse zu den einzelnen Schwerpunkten finden sich im folgenden Abschnitt. Insbesondere wurden aus den Experteninterviews Herausforderungen und Best Practices gewonnen, die den Stand der Praxis wiedergeben, um letzten Endes in Verbindung mit den Ergebnissen der Literaturrecherche Handlungsempfehlungen abzuleiten.

6.2 Übersicht der Befragten

Insgesamt erklärten sich 29 Experten zu einem Interview bereit. Abbildung 37 zeigt die Verteilung der Experten nach Branchen. Insgesamt wurde durch die Experten ein breites Spektrum an Branchen und Tätigkeiten abgedeckt. Insbesondere wurden Nutzer, Werkzeuganbieter, Berater und Forscher befragt. Schwerpunkte zeigen sich bei Werkzeuganbietern, Softwarehäusern und Firmen im Bereich IoT. Während sich das hohe Interesse bei Anbietern auch unter Gesichtspunkten des Marketings erklären lässt, sticht das hohe Interesse bei IoT-Firmen heraus. Dies ist einerseits durch die starke bestehende Vernetzung der Interviewer mit diesen Anbietern zu erklären, andererseits zeigt sich in den Gesprächen, dass diese Firmen in vielen Aspekten bezüglich der Komplexität verteilter IT-Systeme und der Störungserkennung Vorreiter sind.

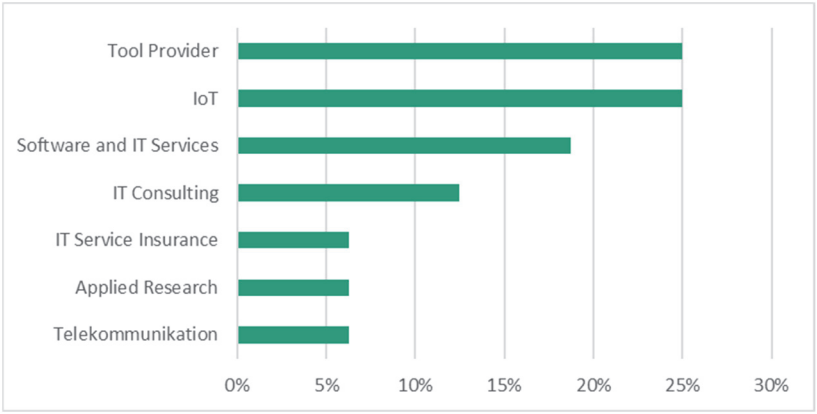

Abbildung 37: Prozentuale Verteilung der Branchen für die Studie

Abbildung 38 zeigt die prozentuale Verteilung der an der Studie beteiligten Unternehmensgrößen. Verschiedene Unternehmensgrößen sind vertreten, von mittelständischen Unternehmen bis hin zu Großunternehmen. Herausforderungen bei der Störungserkennung treten nicht nur in Großunternehmen auf, wobei es sich bei den kleinen Unternehmen vermehrt um Beratungsunternehmen handelt.

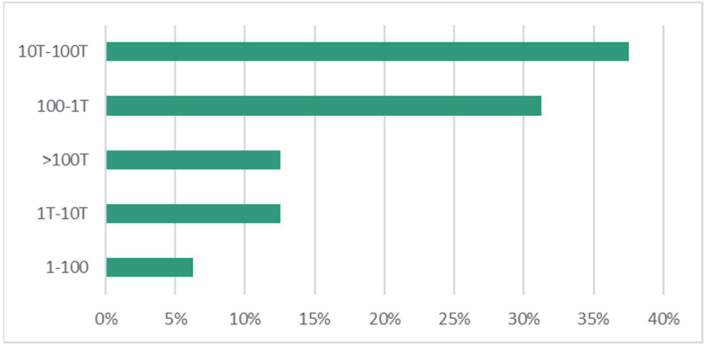

Abbildung 38: Prozentuale Verteilung der Unternehmensgröße für die Studie

Die folgende Abbildung 39 zeigt die Themen der Experteninterviews und die Verteilung der Experten auf diese Schwerpunkte.

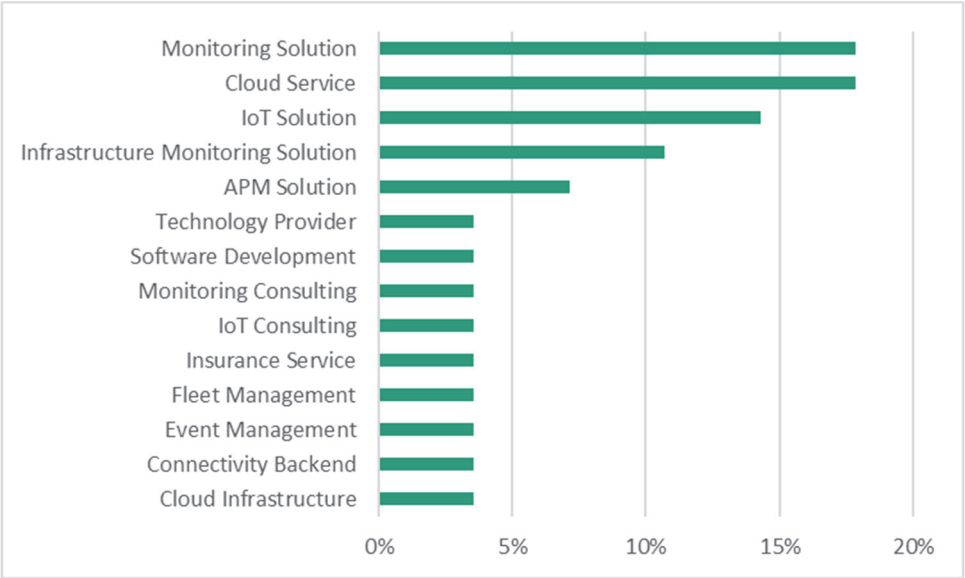

Abbildung 39: Prozentuale Verteilung der Fokusthemen der Unternehmen

6.3 Ergebnisse anhand Schwerpunkten

Im Folgenden werden nun die Ergebnisse der Studie nach Schwerpunkten zusammengefasst. Die folgenden Beschreibungen und Aussagen stützen sich auf die Aussagen und Auswertungen der Interviews.

6.3.1 Spezifikation

Beim Thema Spezifikation geht es grundsätzlich darum, welche Anforderungen an eine Störungserkennung in einem verteilten IT-System gestellt werden. Diese hängen von verschiedenen Faktoren ab, wie beispielsweise vom Kunden, der Kritikalität des Systems, oder den angebotenen Service-Level-Agreements.

Strategie und Ziele. Überwachung und Störungserkennung umfassen technische und organisatorische Herausforderungen. Eine grundlegende Erkenntnis aus den Interviews ist, dass ein Verständnis der Störungserkennung als rein technische Aufgabe zu kurz greift. Bevor eine technische Spezifikation stattfinden kann, sollte eine Unternehmensstrategie zur Überwachung und Störungserkennung festgelegt werden. Diese umfasst alle Teile der Organisation, insbesondere Entwicklung, Betrieb und Support.

Die Spezifikation für eine Störungserkennung sollte sich maßgeblich an der Strategie des Unternehmens orientieren. Um die Strategie zu konkretisieren, werden daraus Ziele abgeleitet. Dabei ist es wichtig, diese Ziele auch messbar zu machen, da diese am Ende auf technischer Ebene gemessen werden müssen.

Ein wesentlicher Orientierungsfaktor ist dabei die Untersuchung, welche Störungen in einem System überhaupt auftreten können. Bei verteilten IT-Systemen (mit zum Beispiel hoher Datenraten) liegen Störungen vermehrt bei »überlasteten« Leitungen als bei Systemen mit beispielweise kleinen Datenraten, die jedoch komplexe Prozesse bzw. Berechnungen durchführen müssen und somit eher Störungen in der Performance aufweisen können. Um für das System die passenden Störungen zu identifizieren, sollten verschiedene Sichten berücksichtigt werden, wie z. B. die technische und fachliche Sicht.

Trennung von fachlicher und technischer Sicht. Die Befragung zeigt, dass eine klare Aufteilung in fachliche und technische Ziele von Vorteil ist. Auf fachlicher Seite steht dabei das »Was« im Vordergrund: Welche Größen sollen gemessen werden, welche Störungen sind relevant, wie sind die Service-Level, etc. Auf technischer Seite steht das »Wie« im Vordergrund: Wie können Werte erhoben oder errechnet werden, wie werden Ziele überwacht und Störungen erkannt, etc.

In der Praxis sollte zwischen beiden Seiten ein Dialog stattfinden, sodass das fachlich Spezifizierte auch technisch machbar ist. Ein Negativbeispiel wäre, Service-Level von fachlicher Seite schon vertraglich festzuhalten, ohne auf technischer Seite geklärt zu haben, ob die Einhaltung überhaupt möglich ist.

Messwerte und Metriken. Aus fachlichen Zielen werden dann Metriken entwickelt, um die Erreichung der definierten Ziele zu messen. Metriken sind in diesem Zusammenhang bei der Störungserkennung Zahlenwerte, um Eigenschaften des Systems zu beschreiben. So ist es möglich das System anhand von Maßzahlen zu bewerten und auch mit anderen System zu vergleichen. Maßzahlen als Metriken können zum Beispiel die Verfügbarkeit des Systems sein, die CPU-Auslastung von Servern, Dauer von HTTP-Requests oder auch die Anzahl von auftretenden Exceptiones innerhalb des verteilten IT-Systems. In diesem Zusammenhang ist es wichtig, die Ziele so zu definieren, dass diese auch messbar sind. Eine Messbarkeit kann direkt oder indirekt gegeben sein. Eine direkte Messung wäre in Bezug auf die Verfügbarkeit die Durchführung eines Pings. Eine indirekte Messung in diesem Zusammenhang wäre die Verfügbarkeit über eine andere Komponente zu messen, beispielsweise an der Performance einer Kundenschnittstelle, die auf die zu messende Komponente zugreift. Eine andere Form der indirekten Messbarkeit ist Berechnung. So kann man beispielsweise aus dem Zeitpunkt einer Anfrage und einer Antwort die Bearbeitungszeit berechnen.

Definition von Metriken. Metriken werden in der Praxis oft aufgrund von Erfahrungswerten festgelegt. Entwickler, Endbenutzer und auch Servicemitarbeiter können hier einen wesentlichen Beitrag leisten, welche Metriken am besten geeignet und auch notwendig sind. Des Weiteren können auch Metriken mittels statistischer Verfahren oder Machine Learning ermittelt werden. Dazu ist es aber notwendig, Vergangenheitswerte zur Verfügung zu stellen. Bei vergangenen Werten ist allerdings Vorsicht geboten, da diese eventuell nicht mehr auf die aktuelle Situation anwendbar sind, beispielsweise, weil der Kundenstamm sich vergrößert hat oder Geschäftsprozesse verändert wurden.

Auch wenn es sich lohnt, anfangs Zeit in die Definition von Metriken zu investieren, ist die Definition von Metriken ein iterativer Prozess. Wird zum Beispiel eine hohe Anzahl falsch positiver Alarme festgestellt, könnte ein Schwellwert gesenkt werden. Wird im Gegenzug dazu festgestellt, dass eine Störung vorlag, ohne dass diese erkannt wurde, könnte ein Schwellwert erhöht werden.

6.3.2 Service-Level

Inhalt von Service-Levels. Service-Levels sind grundsätzlich quantitative Angaben was ein Service (Dienstleistungen) anbieten soll. Beim Thema Störungserkennung in verteilten IT-Systemen beschreiben Service-Levels beispielsweise akzeptable Antwortzeiten, den Datendurchsatz, die Verfügbarkeit oder auch die Fehlerrate, die ein System bei einer Störung haben darf (Beyer, 2018).

Die Praxis zeigt, dass der Umfang des Inhaltes eines Service-Levels zum einen nicht zu grobgranular, jedoch auf der anderen Seite nicht zu feingranular definiert sein soll. Bei einem zu feingranularen Service-Level sind zu viele Bedingungen und Überwachungsprozesse verknüpft, die eine hohe und nicht mehr den Zweck erfüllende Komplexität aufweisen können.

Einführung von Service-Levels. Die Einführung von Service-Levels kann eine Herausforderung darstellen, da das System analysiert und bekannt sein muss, für welches ein Service-Level definiert werden soll. Grundsätzlich ist es wichtig, dass Service-Levels schon bei der Entwicklung von Systemen definiert oder zumindest eingeplant werden. Sind schon historisch gewachsene Systeme vorhanden, muss zunächst sichergestellt werden, welcher Service-Level möglich ist. Wird eine nachträgliche Einführung von Service-Levels angestrebt, können diese in einem iterativen Prozess gemeinsam mit dem Kunden definiert werden. Service-Levels sollten in der Praxis zuerst großzügig definiert werden, damit noch »Luft nach oben« ist. Erst im Laufe der Zeit sollte dann entsprechend konkretisiert werden. In der Praxis ist das Vorgehen allerdings nicht immer so strukturiert. Ein Negativbeispiel ist die willkürliche Festlegung, zum Beispiel nach rein organisatorischen Gesichtspunkten (Arbeitszeiten etc.) oder nach Maßgaben des Auftraggebers (die Service-Level sind so hoch, dass der Kunde den Auftrag erteilt).

Service-Level aus Kundensicht. Der Kunde sieht allerdings Service-Level von einer anderen Perspektive, die sich oftmals nicht mit dem vertraglich festgelegten Service-Level deckt. Wird der vereinbarte Service-Level stets übertroffen, so ergibt sich ein neuer, gefühlter Service-Level. Sinkt die Servicequalität auf den vertraglich vereinbarten Service-Level, so wird dies schon als Störung wahrgenommen. So ist es dem Kunden wichtig, dass eine entsprechende Nachricht bei einer Störung zeitnah erfolgt. Allerdings kann ein zu schnelles Benachrichtigen den gefühlten Service-Level des Systems senken. Wird ein Kunde bei einer schnell behobenen Störung sofort benachrichtigt, so ist die Meldung das einzige, was der Kunde von der Störung bemerkt. Ähnliches gilt für den Lebenszyklus eines Produkts. Erwartet der Kunde eine längere Verfügbarkeit von Diensten, die mit dem Produkt im Zusammenhang stehen, so wird er unzufrieden sein, wenn die vertraglich zugesicherte Angebotsdauer eingehalten wird, aber als zu kurz empfunden wird. Aus Sicht des Kunden gibt es ein System, von dem er eine Antwort erwartet. Aus Sicht des Anbieters gibt es ein verteiltes IT-System mit mehreren Komponenten, die individuelles Fehlerverhalten aufweisen.

Überwachung von Service-Levels. Gibt es keine eigene Überwachung der Service-Levels, resultiert dies in einer »Messung durch Störungsmeldung«. Die Störung wird dadurch bekannt, weil der Kunde sie meldet. Am anderen Ende steht die Messung nachweislicher Funktionalität – Dazwischen liegen Metriken wie Pings und Durchlaufzeiten. In der Praxis zeigt sich, dass bei der Überwachung von Service-Leveln nicht nur eine Unterschreitung verdächtig ist, sondern auch eine Überschreitung. Erfolgt eine Antwort beispielsweise schon nach 10 Millisekunden statt wie sonst 100, ist wahrscheinlich, dass der Prozess nicht durchgeführt wurde, sondern sofort einen Fehler erzeugt hat.

Service-Levels in Zusammenhang mit Dritten. Ein einzelner Service-Level ist oft multiplikativ, also von anderen Service-Levels und somit von anderen Systemkomponenten abhängig. Bei einem solchen Gesamt-Service-Level müssen die »Sub-Service-Levels« auch berücksichtigt und gemessen werden. Herausfordernd wird es beispielsweise jedoch, wenn ein Sub-Service-Level bei einem Drittanbieter liegt und die Verfügbarkeit der einzelnen Services nicht gemessen werden kann, sondern nur die gesamte Verfügbarkeit des Drittanbieters-Services, wie Abbildung 40 zeigt. Dieses verursacht zum einen eine Intransparenz der Störungserkennung, zum anderen lässt es das Gesamtergebnis der Verfügbarkeit zum Kunden hin »schlecht« erscheinen (hier in blau), obwohl die eigenen Services (hier in grün) eine sehr hohe Verfügbarkeitsrate aufzeigen.

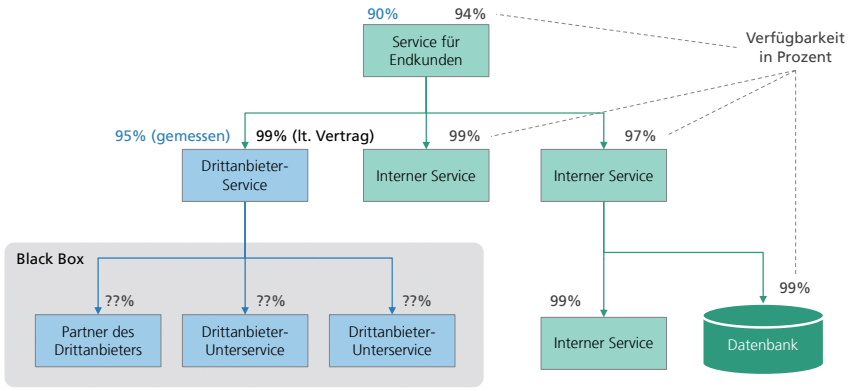

Abbildung 40: Zusammenhang der Verfügbarkeit von Services mit einem Drittanbieter

Sind aus Kundensicht Dritte für den Service-Level verantwortlich, so ist es wichtig, den Umgang mit diesen schon bei Planung des Systems mit zu berücksichtigen. Sonst ist es möglich, dass der dem Kunden zugesicherte Service-Level mit dem Service-Level des Drittanbieters inkompatibel ist. Das Resultat könnte dann sein, dass die Einhaltung des Service-Levels rein rechnerisch nicht möglich ist. Ebenso muss als Teil der Störungserkennung durch Messung auch die Einhaltung des zugesicherten Service-Levels des jeweiligen Dritten sichergestellt werden.

6.3.3 Internet of Things

Die hohe Anzahl an Befragten im Bereich IoT zeigt, dass dieses Thema im Kontext von verteilten IT-Systemen eine immer stärkere Rolle spielt und besondere Herausforderungen für die Störungserkennung mitbringt.

Verteiltes IT-System als IoT. Ein verteiltes IT-System kann generell als IoT betrachtet werden. Ein »Thing« ist in einem verteilten IT-System grundsätzlich nichts anderes als eine Teilkomponente des Gesamtsystems, wobei die Besonderheit ist, dass diese beweglich ist und in der Hand des Kunden sein kann. Dies kann eine bestimmte Hardware sein oder eine spezifische Software (siehe Abbildung 41).

Zu der Hardware zählt beispielsweise ein Server, ein Router ein LAN oder WLAN-Adapter oder aber auch ein Smartphone oder ein Auto. Softwarekomponenten werden in diesem Kontext oft als Services beschrieben, die in sich ein geschlossenes System (unabhängig und modular) darstellen und über das Netzwerk erreichbar sind (Bianco et al., 2007). So kann ein Service zum Beispiel eine Softwarekomponente sein, die den Internetverkehr an einem Server misst, Anfragen an eine Datenbank entgegennimmt oder ein Dienst, der den Datenverkehr in einem verteilten IT-System überwacht.

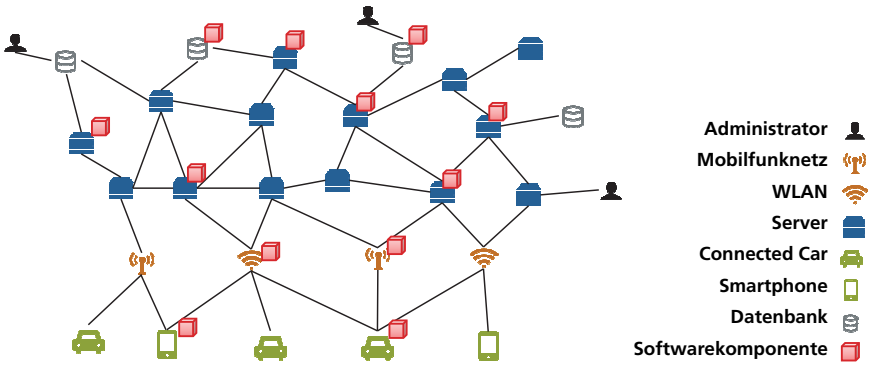

Abbildung 41: Verteiltes IT-System als IoT

Internet of Things verändert die Sichtweise. Mit IoT verändert sich die Sichtweise auf das Gesamtsystem. Die Erbringung einer Dienstleistung ist zwar für den Kunden immer noch ein Produkt, oder Teil eines Produkts, jedoch ist für den Anbieter mit der Auslieferung des Produktes erst ein Teil getan. Services müssen über den Lebenszyklus hinweg betreut werden. Die Qualitätssicherung ist ein kontinuierlicher Prozess geworden, was über die Sicherstellung der Produktfunktion beim Verlassen des Werks hinausgeht und vielmehr den dauerhaften Betrieb betrifft.

IoT steigert die Komplexität. Die Befragung zeigt, dass das Thema IoT die Komplexität in verteilten IT-Systemen stark ansteigen lässt. Verteilte IT-Systeme setzen sich nicht mehr wie früher durch einen Server, einen Desktop, ein Betriebssystem und ein Netzwerk zusammen, sondern bestehen aus einer wachsenden Vielfalt an Komponenten, angefangen von Servern, Betriebssystemen, Datenbanken, Devices (z. B. Smartphones, Autos, Tracker, Uhren, etc.), mobilen Verbindungen, Firewalls und vielem mehr. Ein wesentlicher Faktor, der diese Komplexität noch weiter anwachsen lässt, ist die geforderte Skalierbarkeit des Systems. Dies zeichnet sich durch die Fähigkeit aus, mittels dem Hinzufügen von Hardware- und Softwareressourcen eine höhere Last verarbeiten zu können, ohne, dass die Service-Level darunter leiden. Ein Beispiel sind Sensornetze für Städte, die ihre Technologie so gestaltet haben, dass eine hohe Flexibilität und auch Anpassbarkeit gewährleistet ist. So ist es einfach möglich, Datenquellen oder Sensoren hinzuzufügen. Die Skalierbarkeit erschwert die Störungserkennung in einem verteilten IT-System, falls eine hohe Fluktuationsrate bezüglich der Komponenten vorhanden ist, da es schwieriger ist, normales von abnormalem Verhalten zu unterscheiden.

6.3.4 Datenintegration

Um eine Störungserkennung in einem verteilten IT-System zu etablieren, müssen Daten, über welche Störungen erkannt werden können, gesammelt und integriert werden.

Umsetzung einer Datenintegration. Eine klassische Herangehensweise in der Praxis eine Datenintegration umzusetzen, ist der Einsatz von Monitoring-Systemen, bei denen Daten von allen Unternehmensbereichen eines Systems zusammenlaufen (integriert werden). Eine umfassende automatische Erkennung von Störungen ist nur möglich, wenn Daten aus allen relevanten Bereichen des Unternehmens integriert werden. Dazu gehört neben der Überwachung von Soft- und Hardware auch der Datenverkehr im Netzwerk.

Drittsysteme bei der Datenintegration. Sind Drittanbieter-Systeme in einem verteilten IT-System vorhanden, ist eine Datenintegration schwer, da solche oft als Black Box in das System integriert werden Dies bedeutet, dass lediglich die funktionalen Schnittstellen bekannt sind, Informationen über die innere Funktion allerdings nicht verfügbar sind. Wo möglich, ist im Vorfeld der Vertragsgestaltung festzulegen, welche Daten für die Überwachung von Störungen notwendig sind, sodass der Drittanbieter entsprechende Schnittstellen bereitstellt. Im Allgemeinen sind die Einflussmöglichkeiten auf die Gestaltung von Komponenten Dritter allerdings begrenzt. Viele Anbieter im Cloud-Bereich bieten heutzutage umfassende Schnittstellen zum Monitoring an. Selbst wenn Drittanbieter Daten zur Verfügung stellen, ist es sinnvoll, unabhängig davon eine Überwachung durchzuführen. Nur so lässt sich die effektive Verfügbarkeit feststellen, die auch die Verbindung der eigenen Systeme zur Drittkomponente umfasst.

Die Ende-zu-Ende-Überwachung einer Drittanbieter-Komponente ist immer möglich. Dabei werden ein- und ausgehende Nachrichten zur Drittanbieterkomponente überwacht, woran sich beispielsweise die Antwortzeiten messen lassen. Ebenso ist es möglich, Testanfragen mit bekannten Antworten als Sonden zu verwenden, um ein Black-Box-System funktional zu überwachen.

Testen der Datenintegration. Aus den Befragungen wurde deutlich, dass eine Datenintegration unter realen Bedingungen getestet werden muss. Werden selbst aufgebaute Testsysteme für Ende-zu-Ende-Tests herangezogen, muss jedoch berücksichtigt werden, dass bei einem Life-System andere Sicherheitsrichtlinien oder Firewalls vorhanden sein können, die im Zweifelsfall Nachrichten »verschlucken«. So kann es geschehen, dass eine Störung zwar an sich erkannt wird, der entsprechende Alarm aber nicht das Ziel erreicht.

Um die Datenkommunikation zu testen, sind manuell ausgelöste »Probe-Alarme« ein bewährtes Mittel. So können Ende-zu-Ende-Tests durchgeführt werden, die sicherstellen, dass alle Messdaten ihr Ziel erreichen. Ursachen für nicht angekommene Alarme müssen dann ermittelt und behoben werden.

Minimalinvasiver Ansatz. In den Befragungen empfohlen einige Experten bei Umsetzung der Datenintegration einen minimalinvasiven Ansatz. Dieser verfolgt das Ziel, möglichst wenige Veränderungen im bestehenden System durchzuführen, beispielsweise Freischaltungen von Firewall-Ports oder Gewährung von Rechten. Der Einsatz von Agenten an Hardware- oder Softwarekomponenten birgt das Risiko, Instabilitäten und Netzwerksauslastung im Produktivsystem zu erzeugen. Gerade Deployment und Konfiguration von Agenten wurde von einigen Befragten als Problem hervorgehoben.

Health Endpoints. Health Endpoints sind ein bewährtes Mittel, um Anwendungen nach ihrem »Gesundheitszustand« zu überprüfen. Sie geben Auskunft über die Integrität im Gesamtsystem und werden von der Mehrheit von den Befragten als state-of-the-art angesehen. Abbildung 42 veranschaulicht, welche Möglichkeiten ein Monitoring-Werkzeug hat, eine Anwendung über einen Health Endpoint zu überwachen. Über periodische Abfragen können beispielsweise die Erreichbarkeit, die Integrität oder andere Kennwerte periodisch abgefragt werden. Einerseits dienen die aktuellen Kennwerte dazu, Regeln auszuwerten und gegebenenfalls Alarme zu erzeugen, andererseits bilden sie auch Teile einer Historie, die für Aufgaben wie Visualisierung und Mustererkennung zur Verfügung stehen.

Der Umfang der Überprüfung eines Systems mittels Health Endpoints ist allerdings begrenzt. Da Health Endpoints Teil der Anwendung sind, können sie keine Informationen über spezifische Störungen liefern, die die Anwendung selbst nicht erkennt, verursacht beispielweise durch byzantinische Fehler oder kundenspezifische Fehler. Das gesamte Monitoring kann daher nicht nur mittels Health Endpoints umgesetzt werden, sondern diese dienen nur als Teil der Störungserkennung.

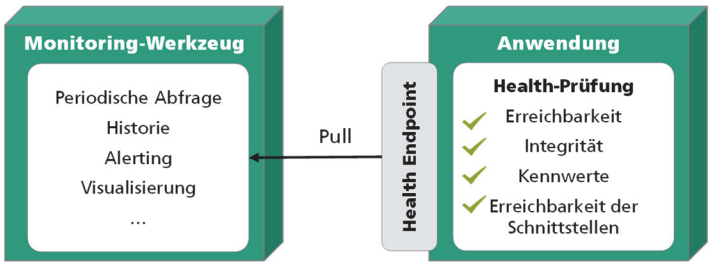

Abbildung 42: Darstellung für die Verwendung eines Health Points für eine Anwendung

Event Hooks. Eine weitere Möglichkeit, den »Gesundheitsstatus« zu überprüfen, sind Event Hooks (oder auch Webhooks). Diese Technologie wird von Befragten auch als Alternative zu Log-Tracing gesehen. Ein Event Hook ist eine Schnittstelle, mittels der eine Komponente Ereignisse einer anderen Komponente abonnieren kann. Umgesetzt wird dies mittels einem Publish/Subscribe-Konzept. Abbildung 43 veranschaulicht die Funktionsweise eines Event Hooks. Über eine Hook URL abonniert das Monitoring-Werkzeug bei der Anwendung über eine Event API spezifische Ereignisse (Events), die übermittelt werden sobald diese auftreten. Das Monitoring-Werkzeug empfängt diese Events dann über eine eigene Schnittstelle (Event Acceptor) und kann diese entsprechend z. B. durch Visualisierungen verarbeiten.

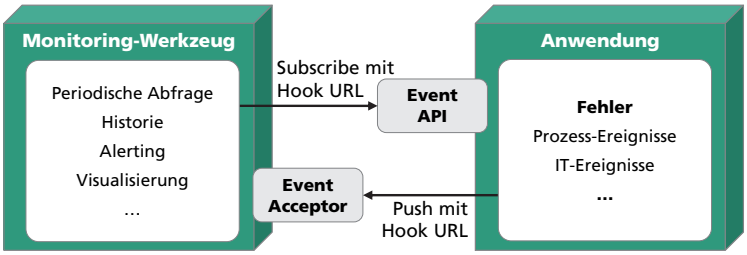

Abbildung 43: Konzept und Funktionsweise eines Event Hooks

Event Hooks können Ereignisse in verschiedenen Formaten wie beispielsweise in JSON oder XML bereitstellen. Auch können Ereignissen nach Typen gefiltert werden (z. B. nur Fehler von Netzwerkprozessen).

6.3.5 Überwachung von Systemen Dritter

Das Vorhandensein von Drittsystemen in verteilten IT-Systemen und deren Überwachung wird bei den Befragten als eine notwendige Herausforderung wahrgenommen. Ein Drittsystem ist ein System, das von einer Partei betrieben wird, die weder zum Dienstanbieter noch zum Dienstabnehmer gehört, beispielsweise einem IT-Dienstleister.

Systeme Dritter als Black Box. Systeme Dritter werden oft als sogenannte Black Boxes wahrgenommen, deren innere Funktion nicht bekannt ist und deren innerer Zustand nicht in Erfahrung gebracht werden kann (siehe Abbildung 44). Natürlich ist eine solche Black Box überwachbar, jedoch lediglich nur am »Rand« über die In- und Outputs (z. B. via Service-Schnittstellen). So kann die Überwachung aus Nutzersicht nur durch spezifische Anfragen, durch Pings oder der Auswertung des Outputs getätigt und Rückschlüsse auf Störungen ermittelt werden. Der Fokus liegt in diesem Falle bei der Gegenüberstellung von In- und Outputs und beobachtet funktionale und zeitliche Aspekte wie Antwortzeiten.

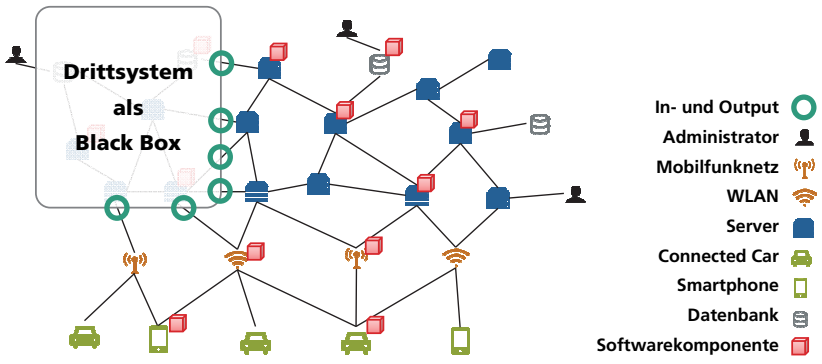

Abbildung 44: Drittsysteme in einem verteilten IT-Systemen als Black Box

Für die Überwachung ist jedoch notwendig, dass der Input nicht selbst fehlerhaft ist, denn in der Datenverarbeitung gilt: »Garbage In, Garbage Out«. Eine Falscheingabe zieht einen Fehler nach sich, der sich bestenfalls in einer Fehlermeldung äußert, aber auch einen Fehler anderer Klassen erzeugen kann, bis hin zum byzantinischen Fehler. Somit ist es notwendig Inputs an Drittsysteme immer zu validieren, um Fehler an der Schnittstelle zu erkennen. Dies gilt auch für die Outputs von Drittsystemen bevor das Ergebnis im eigenen System weitergereicht wird. Hier ist eine Kontrolle natürlich schwierig, da das Drittsystem ja eine notwendige Funktion erfüllt, die eigene Systeme nicht abdecken können. Plausibilitätsprüfungen sind allerdings vergleichsweise einfach zu implementieren und decken viele Fehlerklassen ab.

Strategien zur Überwachung von Drittsystemen. Die Befragten nannten verschiedene Strategien, die sich bei der Überwachung von Drittsystemen bewährt haben. Wie beschrieben, sollten sowohl Inputs als auch Outputs unabhängig überwacht werden. Das bedeutet auch, den Service-Level selbst zu messen und sich nicht auf Angaben Dritter zu verlassen.

Beispielanfragen, für die die korrekte Antwort bereits bekannt ist, können zur Überwachung von Drittsystemen benutzt werden. Werden diese periodisch gestellt, kann bei Abweichungen oder Nichtantwort ein Alarm ausgelöst werden. Wo möglich, können auch Sensoren wie Pings eingesetzt werden, die das Drittsystem aus der Ferne überwachen.

Neben der Funktion sollten auch das Zeit- und Mengenverhalten überwacht werden. Ist ein Eingangskanal still (gehen also keine Nachrichten ein), so ist dies immer ein starkes Indiz für einen Fehler, insbesondere, falls Kundenanfragen über ein Drittsystem in das eigene System gelangen.

Offenlegung von Drittsystemen. Ist es möglich, Drittsysteme offenzulegen, bietet dies viele Vorteile bei der Überwachung und Störungserkennung. Allerdings besteht in den meisten Fällen nur eine sehr begrenzte Einflussmöglichkeit auf die Systeme Dritter, insbesondere, wenn es noch andere Nutzer gibt.

So kann beispielsweise auf die Gestaltung der Komponenten Dritter Einfluss genommen werden, indem den Softwarekomponenten spezifische Software zur Überwachung integriert werden. Alternativ kann die Black Box mit manchen Werkzeugen transparenter gemacht werden. Java Anwendungen bieten z. B. die Möglichkeit sich auf eine Java Virtual Machine »einzuklinken«, sodass ohne Quelltext feingranularer überwacht werden kann. Verständlicherweise wollen Drittanbieter oftmals ihre Lösung nicht oder nur zum Teil offenlegen. Dies kann unter Umständen weitere Kosten verursachen, wenn Anforderungen an die Überwachung eines Drittsystems gefordert werden.

Manche Befragte haben in den Ausschreibungskriterien bzw. bei Vertragsbehandlungen bereits die Anforderungen zur Überwachung berücksichtigt. So kann sichergestellt werden, dass dem verteilten IT-System nur solche Drittsysteme hinzugefügt werden, die auch in das Konzept zur Überwachung und Störungserkennung passen. Gerade im Cloud-Bereich bieten viele Anbieter ihren Kunden bereits Überwachungslösungen als Teil des Produkts. Hier sollte auch auf die Möglichkeit eines Rohdatenzugriffs geachtet werden, da sonst die Servicequalität nicht unabhängig überprüfbar ist.

Eine Herausforderung bei der Offenlegung zwischen Unternehmensbereichen ist das Entstehen von Ad-Hoc-Lösungen. Werden Messdaten, Kennwerte und Werkzeuge nicht geteilt, entstehen Redundanzen. Redundante Systeme erzeugen Mehraufwand und erschweren die Kommunikation während der Störungserkennung, da verschiedene Parteien unterschiedliche Datenbestände haben.

6.3.6 Unternehmensübergreifende Überwachung

Verteilt sich ein IT-System über mehrere Unternehmen, muss die Überwachung ebenfalls unternehmensübergreifend erfolgen. Dies stellt eine besondere organisatorische und technische Herausforderung dar, auch weilunterschiedliche Akteure verschiedene Interessen vertreten.

Umsetzung unternehmensübergreifender Überwachung. Eine unternehmensweite Überwachung hat jedoch einige Voraussetzungen, um effizient zu funktionieren. Aus den Befragungen wurde deutlich, dass dafür eine gemeinsame unternehmensweite Strategie und davon abgeleitete Ziele definiert werden müssen. Dazu gehört, was überwacht werden soll, wie überwacht werden soll, wo die Verantwortlichkeiten liegen und welche Werkzeuge und Standards eingesetzt werden sollen.

Des Weiteren ist eine globale Sicht auf das Gesamtsystem notwendig. Es muss also eine zentrale Stelle geben, bei der eine komplette Datenintegration stattfinden kann, eine Art Kommandozentrale in der alle Fäden zusammenlaufen. So ist es viel einfacher, den Zustand des Gesamtsystems zu visualisieren. Soll keine vollständige Transparenz gewährleistet werden, kann eine Kompromisslösung eingesetzt werden, die nur einen generellen Zustand einzelner Systeme anzeigt, zum Beispiel anhand von Ampeln oder auch von Landkarten. Gerade für das Management und bei Situationen, bei denen es schneller Reaktionen bedarf, bietet ein »Blick-auf-alles« wesentliche Vorteile.

Umgang mit Insellösungen. Besonders herausfordernd kann das unterschiedliche Interesse zwischen dem Erhalten von historisch gewachsenen Insellösungen zur Überwachung und dem Einsatz einer neuen globalen Überwachung sein. Dies zeigte sich auch bei manchen der Befragten. Insellösungen verbergen ein historisches Besitzdenken, gerade was die Herkunft der Daten anbelangt. Eine Veränderung bzw. Ablösung von Insellösungen scheitert oftmals an der Akzeptanz der Beteiligten und auch an einer notwendigen Definition von gemeinsamen Standards.

Kommunikationsfluss und Dokumentation. Auf organisatorischer Ebene ist es wichtig, dass alle Beteiligten die Kommunikationsflüsse kennen. Wer muss über Fehler unterrichtet werden? Wer ist Ansprechpartner für ein System? Welche Prozesse hängen von welchem System ab? In gewachsenen Systemen kann dies eine Herausforderung sein, da die Kommunikationsprozesse meist ebenfalls gewachsen sind und von Individualwissen und persönlichen Beziehungen der Mitarbeiter abhängen. Dies kann beispielsweise beim Weggang von Mitarbeitern oder bei Krankheit zu Problemen führen. Daher empfiehlt es sich, Kommunikationsprozesse zu dokumentieren und angemessene Vertretungsregeln zu etablieren.

6.3.7 Störungserkennung

Basierend auf den Daten der Überwachungen müssen Störungen erkannt werden. Dies kann entweder manuell oder automatisch geschehen. Damit verbunden ist die Ursachenerkennung, mittels der (mögliche) Gründe für eine Störung ermittelt werden können. Aus dem Ergebnis der Befragungen zeigt sich, dass hier ein Bedarf an Verbesserung vorhanden ist, trotz neuer technischer Möglichkeiten.

Definition von Normalfall und Fehlerfall. Die Erkennung einer Störung ist nur in dem Maße möglich, in dem zwischen dem Normalfall und einem Nicht-Normalfall (Fehlerfall) unterschieden werden kann. Daher ist es wichtig, auf allen Ebenen des Systems den Normalfall zu definieren. Eine Definition auf hohen Abstraktionsebenen ist jedoch sehr schwierig, weil viele Faktoren und Perspektiven zu berücksichtigen sind. Dazu zählen beispielsweise eingesetzte Technologien, Art der Betriebssysteme, das Wetter, der Aktienmarkt, Feiertage, Marketingaktionen, und auch natürlich der Kunde. Dieser entscheidet maßgeblich durch seine Erwartungen was »normal« ist, unabhängig von Service-

Level-Agreements und Ähnlichem. Durch das veränderliche Umfeld ist der Normalfall selten konstant. Der Normalfall von heute kann der Fehlerfall von morgen sein.

Schon bei der Entwicklung von Systemen sollte der Normallfall definiert oder zumindest geplant werden. Der Prozessverantwortliche sollte in diese Diskussionen miteinbezogen sein und den Überblick haben. Ebenso sollte festgelegt werden, wie der Normalfall ermittelt werden kann.

Überwachungswerkzeuge liefern Fakten aus den Systemen. Diese werden von Menschen interpretiert, um den Normalfall vom Fehlerfall zu unterscheiden. Mehrwert liefern Werkzeuge durch die Interpretation von Fakten, im besten Fall durch eine automatische Unterscheidung von Normalfall und Fehlerfall.

Erkennung von Störungen. In der Praxis gibt es verschiedene Möglichkeiten, eine Störung mittels der Unterscheidung des Normalfalls und des Nicht-Normalfalls zu erkennen. Der Einsatz von Monitoring-Systemen, Anomalie-Erkennung mittels Machine Learning und auch der Einsatz von Prognosen finden in der Praxis Anwendung.

Eine Möglichkeit, die Funktionalität eines Systems umfassend zu verifizieren, ist End-to-End-Monitoring aus Kundensicht, beispielsweise mit Agenten oder »Robotern«. Diese führen periodisch Aktionen im System durch. In Abbildung 45 ist der Aufbau gezeigt. Dazu werden im System Testnutzer und Datensätze angelegt, die einem realen Kunden entsprechen. Für diesen Testnutzer werden periodisch Prozesse (z. B. Login, Buchung oder eine Anfrage) durchgeführt, um Fehler aufzudecken. Welche Prozesse und Parameter Verwendung finden, werden wie bei Systemtests so festgelegt, dass eine möglichst hohe Abdeckung von Funktionalität gegeben ist. Dadurch kann die ordnungsgemäße Funktion eines Prozesses sichergestellt und eine tatsächliche Verfügbarkeit gemessen werden. Im Vergleich messen andere Verfahren nur theoretische Verfügbarkeit, die aus Prüfungen wie Pings und der Abwesenheit von Fehlermeldungen errechnet wird.

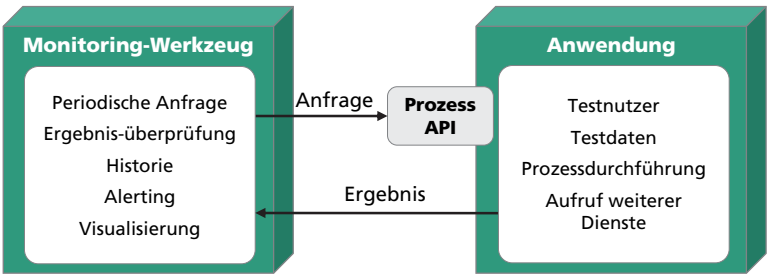

Abbildung 45: Roboter-basiertes End-to-End-Monitoring

Prognosen sind ein aufkommendes Mittel zur Störungserkennung, denn durch eine Voraussage können Störungen erkannt und verhindert werden bevor sie auftreten. So können schon beispielsweise Alarme bei androhenden Regelverletzungen abgesetzt werden. Für Prognosen werden statistische Verfahren und vereinzelt schon Ansätze des Machine Learning eingesetzt. Vereinzelt unterstützen Werkzeuge bereits Prognosen, diese Technologie befindet sich allerdings noch in der Entwicklung.

Für die Anomalie-Erkennung kann ebenfalls Machine Learning eingesetzt werden. Anstatt einer manuellen Definition des Normalfalls wird dieser Anhand von Messdaten automatisch erlernt. Ein Beispiel wäre, dass eine hohe Last auf einem System um Mitternacht als Normalfall gelernt wird, da auf diesem System jede Nacht um 0:00 Uhr ein Backup-Prozess läuft. Die gleiche Last zu einem anderen Zeitpunkt wäre allerdings eine Anomalie. Im Vergleich zu Prognosen bezieht sich die Anomalie-Erkennung nicht auf die Zukunft, sondern auf die Gegenwart. Mehrere Anbieter gaben in Interviews an, Verfahren zur Anomalie-Erkennung bereits zu unterstützen.

6.3.8 Ursachenerkennung

Trotz technischer Neuerung wünschen sich die meisten Befragten Verbesserungen im Bereich der Störungserkennung, insbesondere der Ursachenerkennung. Teilweise wurden auch komplexe Störungen beschrieben, deren Ursache auch nach langer Untersuchung weiterhin unbekannt ist. Dies deutet darauf hin, dass die erhöhte Komplexität verteilter IT-Systeme nicht nur Softwarelösungen zur Störungserkennung, sondern auch Experten an die Grenzen ihrer Fähigkeiten bringt.

Stand der Technik. Vereinzelt sind Lösungen zur automatisierten Ursachenerkennung vorhanden, zum Beispiel für Netzwerkstörungen. Auf Anwendungsebene ist dies allerdings überwiegend nicht der Fall. Für die Ursachenerkennung werden typischerweise Log-Dateien eingesetzt, weil sie einen vollständigen Blick auf das verteilte IT-System bieten. Im Zweifelsfall werden Entwickler hinzugezogen, die Fehler in den Anwendungen auf tiefer Ebene nachvollziehen können. Eine Methode zur Vereinfachung der Ursachenerkennung ist die Verwendung von automatisierten Selbsttests, die Indizien auf die Ursache liefern können.

Auch unterstützen Werkzeuge die Ursachenerkennung durch Korrelationsanalyse. Dabei werden Korrelationen zwischen verschiedenen Zeitreihen von Messwerten identifiziert. Zum Beispiel kann eine Korrelation zwischen der Anzahl der Datenbankzugriffe und der Nicht-Erreichbarkeit der Webseite darauf hinweisen, dass die Webseite nicht erreichbar ist, wenn die Datenbank überlastet ist. Abhilfe könnte dann zum Beispiel eine Aufrüstung des Datenbankservers schaffen.

6.3.9 Standardisierung

Die Etablierung von Standards in verteilten IT-Systemen steht eine immer stärker wachsende Fülle an Schnittstellen, Technologien, Datentypen und Hard- und Softwarekomponenten gegenüber. Trotz der Notwendigkeit nach der Existenz von Standards ist der Grad der Standardisierung gering, beziehungsweise werden viele konkurrierende Standards gleichzeitig eingesetzt. Dies wird auch von den Befragten bestätigt.

Werkzeuge begegnen dieser Herausforderung durch eine große Zahl von Adaptern, Plug-Ins, etc. mittels denen sich quasi beliebige Datenquellen anbinden lassen. Über Datenformate hinaus ist allerdings eine Standardisierung der Semantik von Messdaten ebenfalls wichtig. Ein Beispiel aus der Praxis ist die Etablierung von Zustandsmeldern in allen Systemen. Dabei sollte jedes System eine Ampelfarbe melden (rot, gelb, grün). Allerdings war im Vorhinein nicht standardisiert, was genau die einzelnen Farben bedeuten. Insbesondere beim Zustand Gelb gab es weit abweichende Interpretationen. Bei komplexeren Messwerten kann ein solcher Mangel an Standardisierung ebenfalls zu Problemen führen, beispielsweise bei Durchlaufzeiten und Verfügbarkeiten. So entsteht ein Risiko, »Äpfel mit Birnen« zu vergleichen. In der Praxis steht bei Eigenentwicklungen von Software oftmals die Funktion im Vordergrund, ohne dass auf standardisierte Schnittstellen geachtet wird.

Standards bei Fehlermeldungen mit Fehlercodes.
Zur Identifizierung und Kommunikation von Fehlern sind eindeutige Fehlercodes hilfreich. In kommerziellen Produkten werden Fehler oftmals anhand von Fehlercodes identifiziert, die einem herstellerspezifischen Standard entsprechen. Grundsätzlich gibt es keine übergreifenden Standards für Fehlercodes, sondern diese sind immer spezifisch für einzelne Komponenten, Protokolle (z. B. HTTP, FTP, TCP, oder ICMP) oder Anwendungen (Software) definiert (z. B. SQL). Protokoll-spezifische Fehlercodes helfen beispielsweise, Netzwerk-Fehler in vielen Szenarien in verteilten IT-Systemen zu identifizieren. Zusätzlich können Web-Anwendungsserver wie Tomcat, Glassfish oder Apache so konfiguriert werden, dass sie interne Fehlermeldungen als Antwort über HTTP mit einem Fehlercode zurückgeben, wie Abbildung 46 zeigt.

Abbildung 46: Apache Tomcat[6] Fehlermeldung mit Fehlercode 500

In der Dokumentation gibt es für jeden Fehlercode eine Beschreibung, die besagt, wann dieser Fehler auftritt und im Idealfall auch Hinweise zu Ursachen und Fehlerbehebung gibt. Bei der Entwicklung von firmeneigenen Anwendungen kann es sinnvoll sein, ebenfalls Fehlercodes zu verwenden, die ggfs. auch firmenübergreifend standardisiert sind. Ein hierarchisches System, vergleichbar zu den OIDs in SNMP, kann helfen, Fehler auf Verantwortliche und Systeme zuzuordnen. So kann auch vermieden werden, dass Fehlercodes doppelt vergeben werden. In der Softwareentwicklung erfordern Fehlercodes allerdings eine höhere Disziplin, da Entwickler nicht einfach Fehlermeldungen programmieren können, sondern einen bestehenden oder neuen Fehlercode verwenden und dokumentieren müssen.

Standards im Bereich Exception Policies. Sind erst einmal Fehler aufgetreten, müssen diese auch behandelt werden. Dafür gibt es Exception Policies, die festlegen wie mit Fehlern umgegangen werden soll. Dazu gehören die Fragestellungen welche Fehler von der Anwendung weitergegeben (geworfen) werden dürfen, welche Fehler behandelt werden müssen und wie die Fehlermeldungen an verschiedenen Schnittstellen aussehen sollen. Eine Fehlermeldung in Form eines Fehlercodes mit Stacktrace (wie in Abbildung 46) kann beispielsweise für automatisch angesprochene Schnittstellen sinnvoll sein, da hier umfassende technische Informationen für die Ursachenerkennung übergeben werden. Für eine Benutzerschnittstelle wäre eine solche Fehlermeldung allerdings nicht geeignet, da sie für einen Laien nicht verständlich und auch schwer an den Anbieter kommunizierbar ist.

Logging Policies als Logging Standards. Solche Standards definieren den Aufbau von Log-Nachrichten und Log-Formaten. Eine unternehmenseinheitliche Logging Policy legt grundlegend fest, was in welcher Form auf welchem Log-Level (z. B. info, warning, oder error) geloggt wird. So können Daten strukturiert ins Log geschrieben werden (z. B. als JSON oder XML), anstatt dass der Entwickler selbst einen Fehlertext mit Teilen der Daten generiert. Ein Befragter gab an, mit jedem Log-Eintrag die ID der aktuellen Anfrage zu loggen, was zwar bei nachträglicher Programmierung einen hohen Aufwand mit sich brachte, die Ursachenerkennung aber stark vereinfache. Eine anderer Befragter gab als Beispiel an, Exceptions grundsätzlich mit einem Stacktrace zu loggen, der genau angibt, wo die Exception auftrat.

Logging Policies dienen dazu, die Nutzbarkeit des Loggings für den Betrieb und die Störungserkennung zu erhöhen. Logging dient Entwicklern zunächst als Werkzeug während der Softwareentwicklung. Ohne Standards besteht die Gefahr, dass Logging nur für die Belange der Entwicklung implementiert wird. In solch einem Fall ist die Verwendung von Logs für die Störungserkennung nur eine sekundäre Funktion.

[6] https://tomcat.apache.org

6.3.10 Anpassung und Erweiterung

Eine etablierte Störungserkennung muss flexibel anpassbar und erweiterbar sein, da sich verteilte IT-Systeme ändern und damit auch Störungen verändern und hinzukommen können. Abbildung 47 zeigt einen Prozess für den Umgang mit Änderungen, angelehnt an den Software-Lifecycle.

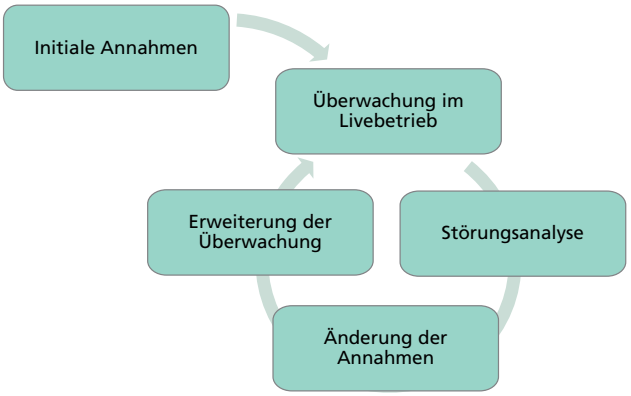

Abbildung 47: Prozess zur stetigen Anpassung und Erweiterung einer Störungserkennung

Zunächst werden während der Entwicklungen initiale Annahmen getroffen, die unter anderem das Mengengerüst, Fehlerverhalten und relevante Messwerte, Daten und Funktionen zur Störungserkennung umfassen. Basierend auf diesen initialen Annahmen bzw. Definitionen von möglichen auftretenden Störungen wird ihr mögliches Auftreten im Livebetrieb überwacht. Tritt eine Störung auf bzw. besteht der Verdacht, dass eine Störung aufgetreten ist, wird eine Störungsanalyse durchgeführt. Diese beinhaltet zum einen was für eine Ursache diese Störung hatte und zum anderen, ob Änderungen in den Annahmen bzw. in den Definitionen gemacht werden müssen. Ist eine neue Störung hinzugekommen, wird diese zu den Annahmen hinzugefügt. Basierend auf den geänderten Annahmen muss die Überwachung im Livebetrieb dahingehend erweitert werden. Wird festgestellt, dass keine Störung vorlag, müssen eventuell Obergrenzen für Alarmregeln angepasst werden. Trat ein Softwarefehler auf, muss dieser behoben werden. Wurde eine bisher unbekannte Störungsursache entdeckt, muss evtl. eine neue Regel oder ein neuer Test der Überwachung hinzugefügt werden. Ebenso kann es sinnvoll sein, die nicht-technischen Teile der Überwachung anzupassen, z. B. die Dokumentation oder die Kommunikationsprozesse zwischen Stakeholdern. Für diesen Prozess kann ein geeignetes Softwarewerkzeug wie ein Ticketsystem verwendet werden.

Mehrere Befragte gaben an, dass es jedoch nur schwer möglich ist, die Überwachung eines Systems auf Anhieb korrekt zu umzusetzen. Statt aufwändig genaue Schwellwerte, notwendige Alarme etc. genau zu definieren, sollte Software so gestalten sein, dass sich diese Aspekte einfach ändern lassen. Die initialen Annahmen können dann als »Best Effort« umgesetzt werden, wobei ein Befragter angab, am Anfang lieber zu viel als zu wenig zu überwachen. Im Livebetrieb wäre es dann möglich unnötige oder redundante Alarme, Fehlermeldungen etc. zu beseitigen oder bei Bedarf neue hinzuzufügen. Bei der Wahl von Softwarewerkzeugen sollte dann demzufolge auf eine einfache Änderbarkeit und Erweiterbarkeit Wert gelegt werden.

6.3.11 Entwicklungsprozess von Softwarekomponenten

Der Versuch eine Störungserkennung erst gegen Ende in einen Entwicklungsprozess einer Softwarekomponente zu integrieren, bedeutet einen erheblichen Mehraufwand. Daher sollte eine Störungserkennung nach der Mehrheit der Befragten schon früh im Entwicklungsprozess von Softwarekomponenten berücksichtigt werden. Der Einsatz von Prozessverantwortlichen bei der Spezifikation und Abnahme hilft, Anforderungen und Ziele im Softwareprozess zu verankern.

Bedeutung der Störungserkennung. Die Integration einer Störungserkennung wurde von vielen Befragten primär als organisatorische Herausforderung eingestuft, da eine unternehmensweite Zusammenarbeit notwendig ist. Geeignete Technologien existieren, allerdings müssen diese auch eingesetzt werden.

Oft scheitert es an dem Bewusstsein, welche wichtige Bedeutung eine Störungserkennung hat (Niedermaier et al., 2019). Dazu gehört auch eine mangelnde Wertschätzung, die sich in fehlender Bereitschaft zeigt, Zeit und Geld zu investieren. Ein oft in der Praxis beobachtetes Phänomen ist ein reaktives Handeln nach einer Störung. Ein Anbieter berichtete, dass es oft erst zum Vertragsabschluss kommt, wenn ein wichtiges System nicht mehr funktioniert und eine Ursachenerkennung nicht möglich ist. Im Vergleich zu einer frühzeitigen Implementierung entstehen so nicht nur Mehraufwände, sondern oft auch ein weniger konsistentes und standardisiertes System zur Störungserkennung.

Als Gegenbeispiel erklärte ein Befragter, dass im Unternehmen die Störungserkennung als gleichwertig zu der eigentlichen Funktionalität der Software gesehen wurde und dementsprechend ähnliche Abnahmeprozesse für Überwachbarkeit notwendig sind, bevor eine Softwareversion in den Betrieb geht. So ist eine hohe Qualität der Störungserkennung durchsetzbar.

Zielgruppen der Werkzeuge. Um die Störungserkennung in einem Entwicklungsprozess von Softwarekomponenten zu fördern, sollte die Bedeutung der Störungserkennung im gesamten Unternehmen hoch sein. Dies beginnt mit einer Festlegung einer Strategie durch die Verantwortlichen, an denen sich alle im Prozess Beteiligten orientieren können. Überwachungswerkzeuge werden von IT-kundigen Entwicklern implementiert, aber von fachkundigen Operations-Experten eingesetzt. Diese Zielgruppe sollte bei der Entwicklung im Auge behalten werden und es sollte regelmäßig zwischen beiden Gruppen kommuniziert werden. Ein Befragter erklärte, Entwickler sollten sich als Werkzeugmacher verstehen, die Operations das Handwerkszeug zum Betrieb geben. Dies kann sogar so weit gehen, dass neben dem eigentlichen Kunden auch Operations eine Endabnahme der Software durchführt.

Entwicklungsaufwand. Die Umsetzung von Überwachbarkeit, Monitoring-Schnittstellen, Logging/Exception Policies und ähnlichen Maßnahmen zur Störungserkennung erfordert wie jede andere Funktionalität in einer Software Aufwand. Im Gegensatz zu der eigentlichen Software-Funktionalität ist hier ein Nutzen nicht unmittelbar sichtbar, weswegen in der Praxis manchmal die Bereitschaft fehlt, den notwendigen Aufwand zu investieren. Vor allem angesichts von Deadlines und Feature-orientierten Kunden, drohen solche nicht-funktionalen Anforderungen wegzufallen. Dies stellt allerdings eine technische Schuld dar, die später mit Mehraufwand zurückgezahlt werden muss.

6.3.12 Störungserkennung im Betriebsprozess

Interne Organisation. Die Störungserkennung sollte im Betriebsprozess verankert sein. Klare Zuständigkeiten und Strukturen sind wichtig. Dazu gehören unter anderem klar definierte Eskalationsstufen, Service-Level, konkrete Ansprechpartner für verschiedene Komponenten und auch klare Informationswege zwischen Kunde und Entwickler. Ohne Informationsweitergabe werden einzelne Mitarbeiter in der Kette zum Ziel für Unmut,

da Sie Prozesslücken nicht überbrücken können. Dies tritt beispielsweise ein, wenn der Kundensupport keine Möglichkeit hat, Störungen zu eskalieren. Deswegen sollte auch ein interner Kommunikationsweg zwischen Entwickler und Kunde vorhanden sein. So weiß beispielsweise der Kundendienst, dass an der Entstörung gearbeitet wird und kann diese Information so an den Kunden weitergeben.

Gemeinsame Ziele und Prioritäten. Verantwortung hierbei liegt beim Management, gemeinsame Ziele und Prioritäten festzulegen und zu kommunizieren. So kann verhindert werden, dass die Priorität von Störungen bei der Entwicklung niedrig, aber beim Support die Priorität hoch eingestuft ist und bei diesem dadurch dann die »Leitungen glühen«. Ein wichtiges, gemeinsam zu kommunizierendes Ziel sollte eine klare Verteilung der Verantwortlichkeiten sein. Wer ist für was zuständig, insbesondere, wer kommuniziert mit dem Kunden? Ein Negativbeispiel aus der Praxis ist die Klärung der Schuldfrage statt die Behebung der Störung. Schuldzuweisungen sind in diesem Zusammenhang kontraproduktiv und können dafür sorgen, dass gegenseitige Einblicke in Daten nicht gewährt bzw. Probleme »unter den Teppich gekehrt« werden.

Reaktion auf Störungen. Die Geschwindigkeit der Reaktion auf eine Störung im Betriebsprozess ist ein sehr bedeutsamer Faktor, um Störungsfolgen zu minimieren. In diesem Zusammenhang müssen auch Meldepflichten oder Fristen in Compliance-Regeln eingehalten werden. Insbesondere bei sicherheitsrelevanten Störungen kann eine frühe Erkennung die Folgen minimieren.

Im schlechtesten Fall ist der Kunde die erste Instanz, bei der eine Störung erkannt wird. Ein Befragter nannte dies »Kunde als Sensor«. Im Idealfall wird eine Störung erkannt und behoben, bevor diese den Kunden überhaupt erreicht.

Zu der Reaktion einer Störung gehört jedoch auch ein Abschätzen der Störungsfolgen bzw. Störungsauswirkungen. Dabei sind folgende Fragen zu klären:

- Wer ist wie stark betroffen? Wer muss benachrichtigt werden?
- Wie lange hält die Störung voraussichtlich an?
- Was sind die Auswirkungen der Störung?
- Wer ist noch betroffen, wenn eine Komponente ausfällt?

Störungen bei Dritten. Eine Störung kann auch außerhalb der eigenen Systemgrenzen liegen, beispielsweise bei einem Dienstleister, Telekommunikationsanbieter, Geschäftspartner oder Kunden. Hier kann es dennoch sein, dass der Kunde sich an das eigene Unternehmen wendet. In diesem Fall ist es hilfreich, Ansprechpartner bei Dritten zu kennen, sodass eine Störungsmeldung erfolgen kann oder der Kunde weitervermittelt werden kann. Im Gegenzug müssen eingehende Störungsmeldungen von Dritten auch im eigenen Betrieb bekannt sein, sodass dem Kunden eine richtige Auskunft gegeben werden kann und doppelte Meldungen vermieden werden können.

Ein Befragter gab an, dass gute Überwachungswerkzeuge Störungen beim Partner erkennen, bevor es der Partner oder die Kunden erkennen. Hier zahlt es sich aus, nicht davon auszugehen, dass die Störung bereits bekannt ist, sondern diese proaktiv zu melden.

6.4 Aktuelle Herausforderungen

Neben dem aktuellen Stand der Praxis wurden die Experten auch befragt, welches aus ihrer Sicht die größten Herausforderungen in der Störungserkennung sind. Wie in Abbildung 48 gezeigt, sind dies (1) Einsatz von Insellösungen, (2) Vielfältigkeit der Komponenten, (3) Vielfalt der Daten und (4) keine Standards für Monitoring-Lösungen.

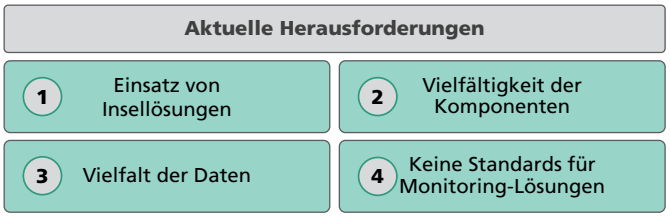

Abbildung 48: Überblick über aktuelle Herausforderungen bei verteilten IT-Systemen

Einsatz von Insellösungen. Insellösungen sind die am öftesten genannte Herausforderung. Historische Entwicklungen von Systemen, die sich im Laufe der Zeit in den Abteilungen und im Alltag der Mitarbeiter und Führungskräfte etabliert haben, sind oft schwer abzulösen. Besonders angesichts der Tatsache, dass solche Insellösungen stark in den Prozess eines Unternehmens integriert sein können und sich auch schon über einen großen Zeitraum bewährt haben. Ein damit verbundenes »Lagerdenken« führt zu Intransparenz, vor allem zwischen den Abteilungen, und verursacht viele organisatorische Hürden und Herausforderungen. Eine Ersetzung bzw. Ablösung solcher Altsysteme ist daher zum einen mit einer geringen Akzeptanz der Benutzer verbunden und auch zum anderen mit hohen technischen und arbeitsintensiven Aufwänden. Dies gilt insbesondere, falls zwei konkurrierende Systeme bestehen, von denen eines in das andere integriert werden soll.

Vielfältigkeit der Komponenten. Eine weitere, weit verbreitete und immer stärker wachsende Herausforderung ist die sich stetige verändernde Vielfalt der in einem verteilten IT-System vorhandenen Komponenten. Dazu zählen beispielsweise Hardware- und Softwarekomponenten, Endgeräte, Werkzeuge, Schnittstellen oder Protokolle. Das Ersetzen, das Hinzufügen oder das Entfernen von solchen Komponenten bringt eine konstante Dynamik mit sich. Durch die Abhängigkeit zwischen Komponenten in verteilten IT-Systemen bringt jede Änderung auch das Risiko einer Inkompatibilität mit sich, selbst wenn die Komponente als solche fehlerfrei funktioniert.

Ein modernes Mittel, dieser Komplexität entgegenzutreten, ist Virtualisierung, z. B. durch virtuelle Maschinen oder Containersysteme wie Docker. Alle Schichten unter der Software sind abstrahiert und werden im Idealfall automatisch verwaltet. Während dies im Regelbetrieb große Vorteile bringt, kann dies bei der Störungserkennung eine zusätzliche Herausforderung sein, da unklar ist, was unterhalb der Software konkret passiert.

Vielfalt der Daten. Die Vielfältigkeit der Komponenten in einem verteilten IT-System versursacht auch eine Fülle an Daten in verschiedenen Formaten. Diese heterogene, immer mehr wachsende Datenmenge erschwert die Störungserkennung. Um einen verteilten Prozess nachzuvollziehen, sind Daten aus mehreren Quellen notwendig. Oft fehlen geeignete Kontextinformationen zu den erzeugten Daten, z. B. der Kontext eines Fehlers, eines Messwertes oder eines Aufrufs. In diesem Zusammenhang ist auch Skalierbarkeit von Lösungen eine Herausforderung, da die Datenmenge ständig wächst.

Keine Standards für Monitoring-Lösungen. Monitoring-Lösungen gehören heutzutage zu den am meisten eingesetzten Überwachungswerkzeugen für Störungen. Jedoch fehlen grundsätzliche Standards für Schnittstellen, Datenformate und Abläufe. Dabei sind Industriestandards gemeint, aber auch firmeninterne Standards. Das Fehlen von beiden erschwert die Integration von Messdaten und Monitoring-Lösungen.

6.5 Zukünftige Entwicklungen

Aus den Befragungen wurden vier Themenbereiche identifiziert, die die Befragten im Rahmen zukünftiger Entwicklungen diskutiert haben. Wie Abbildung 49 zeigt, geht es um (1) Künstliche Intelligenz, um (2) Microservices, um (3) IT-Automatisierung und schließlich um die (4) Bewertungen von Technologien. Im Folgenden wird gezeigt, welche Meinungen zu diesen Themen in den Befragungen gesammelt wurden.

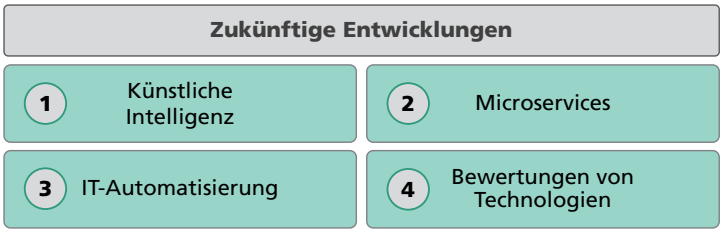

Abbildung 49: Themenüberblick über zukünftige Themen

Künstliche Intelligenz. Künstliche Intelligenz (KI) ist wohl das meist diskutierte Thema, wenn es um Zukunftsthemen im Bereich von Störungserkennung geht. Trotz der starken allgemeinen Innovations- und Zukunftsperspektive wird KI im Bereich der Störungserkennung in verteilten IT-Systemen bei den Befragten sowohl optimistisch als auch skeptisch gesehen. Machine Learning wird von manchen Befragten zur Anomalie- und Ursachenerkennung als sinnvoll erachtet. Dieser Einsatz von Machine Learning zählte für manche Befragte allerdings nicht als »wahre« KI. Das mag auch darin begründet liegen, dass in der Literatur verschiedene Definitionen von KI existieren, die sich zum Teil erheblich voneinander unterscheiden.

Allgemein wird die Rolle der KI in Zukunft eher als eine Ergänzung und Unterstützung bei der Störungserkennung gesehen, daKünstliche Intelligenz erst einsatzfähig ist, wenn Vergangenheitsdaten vorhanden sind, anhand derer gelernt werden kann. Dies setzt aber ein stabiles System mit einem sich nicht ändernden Soll-/ Normalzustand voraus, was in dynamisch verteilten IT-Systemen nur bedingt gegeben ist. Ein Werkzeuganbieter nannte weiterhin die Unterdrückung von Fehlalarmen als Mittel gegen Alert-Fatigue ein Einsatzgebiet für KI, das momentan erforscht wird.

Microservices. Der Bedarf an Dynamik und Skalierbarkeit in IT-Systemen führt zu neuen Technologien und Architekturmustern, die diesen gerecht werden. So erlangen solche Lösungen ein flexibles Design, um auch Änderungen von Anforderungen schneller umsetzen zu können. Ein Beispiel dafür zeigen auch (Lehmann and Freymann, 2018) im Bereich von Smart Cities. Microservices gehören in diesem Zusammenhang zu den state-of-the-art Design-Patterns (Freymann et al., 2020). Sie sind (vergleichsweise) kleine Software-Einheiten, die eine ganz bestimmte Aufgabe erfüllen. Sei es eine Log-Aufgabe, die Validierung einer Eingabemaske, die Speicherung von Daten oder die Nutzerverwaltung. Der Vorteil, ein System in solch kleine Einheiten zu zerlegen, ist die voneinander unabhängige Skalierung dieser Einheiten.. Mit einer monolithischen Software ist dies nicht so einfach möglich. Um diese Microservices effizient zu hosten und den Einrichtungsaufwand zu minimieren, werden Containerlösungen wie Docker eingesetzt. Von einer Mehrheit der Befragten werden solche Containerlösungen als nächste Stufe der Cloud-Infrastruktur nach virtuellen Maschinen gesehen.

Die Störungserkennung in Microservice-Architekturen ist allerdings schwieriger, da durch die Verkleinerung der einzelnen Software-Einheiten die Anzahl an unabhängig voneinander deployten und betriebenen Software-Instanzen wächst. Da Microservices feingranularer sind, muss das, was früher mit Logging möglich war, nun mit Werkzeugen für verteilte IT-Systeme gelöst werden. Werkzeuge zur Überwachung und Störungserkennung müssen mit dieser steigenden Komplexität und Dynamik umgehen können.

IT-Automatisierung. Im Licht der immer weiter wachsenden Menge an zu verwaltenden Systemen spielt die Automatisierung von IT-Aufgaben eine immer größere Rolle. In der Störungserkennung sind dies Aufgaben wie die Integration neuer Systeme in Monitoring-Lösungen und die Erstellung von Systemübersichten. In dynamischen, vernetzten IT-Systemen stoßen manuelle Konfigurationsregeln an ihre Grenzen. Erste Ansätze zur Automatisierung bestehen bereits. So bieten manche Werkzeuge an, Überwachungsregeln für einen Server automatisch zu erstellen, indem dieser auf relevante Software- und Hardwarekomponenten hin gescannt wird. Manuelle Konfiguration ist nur noch nötig, falls von den automatisch erstellten Regeln abgewichen werden soll.

Mögliche zukünftige Entwicklungen könnten Agenten sein, die bei erkannten Störungen selbstständig Standardmuster zur Behebung anwenden oder Alarme versenden. Zukünftige Entwicklungen in diesem Bereich wurden von den Befragten als notwendig und wahrscheinlich eingestuft.

Bewertungen von Technologien. Aufgrund der Fülle der angebotenen Technologien im Bereich der Störungserkennung waren nicht alle Befragte mit jeder Technologie vertraut. Im Folgenden werden einige Experteneinschätzungen wiedergegeben. Das Simple Network Management Protocol (SNMP) für die Überwachung von Netzwerkkomponenten wird eher als Legacy-Technologie angesehen, obwohl es bei vielen Software-Lösungen eingesetzt wird. APIs und Webhooks sind aktuellere Alternativen für den gleichen Anwendungszweck. Bezüglich Open-Source-Software gab es gemischte Meinungen. Ein Befragter gab zu bedenken, dass viele Open-Source-Lösungen das Resultat gescheiterter Startups seien, was nicht gerade für die Qualität und Zukunftssicherheit spräche. Bei der Auswahl von Open-Source-Lösungen sei dementsprechend Vorsicht geboten.

Laut Einschätzung der Befragten wird das Application Performance Management in Zukunft eine Schlüsselrolle einnehmen, insbesondere in unternehmensübergreifenden verteilten IT-Systemen.

7 Best Practices und Handlungsempfehlungen

Dieses Kapitel beschreibt die in den Experteninterviews identifizierten Best Practices. Die Handlungsempfehlungen wurden in sechs Bereiche aufgeteilt, wie Abbildung 50 zeigt.

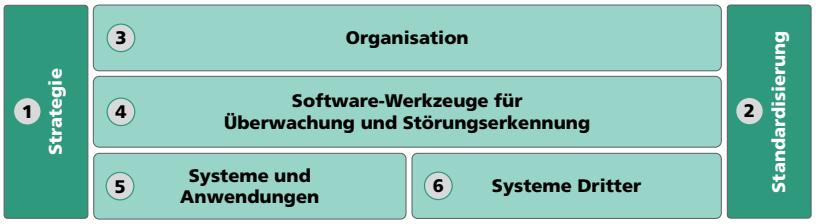

Abbildung 50: Überblick über die Bereiche der Handlungsempfehlungen

Die Unternehmensstrategie (1) für Überwachung und Störungserkennung bildet die organisatorische Basis für alle weiteren Handlungsempfehlungen. Dem gegenüber gestellt steht die Standardisierung (2), die die technische Basis für die Handlungsempfehlungen bildet. Ohne Strategie und Standards ist ein unternehmenseinheitliches Handeln nur schwer möglich.

Die Organisation (3) umfasst alle nicht-technischen Prozesse innerhalb und außerhalb des Unternehmens. Dazu gehören Entwicklungs-, Support- und Störungsbehebungsprozesse. Weitere Handlungsempfehlungen betreffen die Auswahl, Konfiguration und Einsatz von Software-Werkzeugen für Überwachung und Störungserkennung (4). Damit verbunden sind Empfehlungen für Entwicklung, Betrieb und Integration von (Produktiv-)Systemen und Anwendungen (5) und Empfehlungen für die Anbindung von Systemen Dritter (6).

7.1 Strategie

Störungserkennung hat neben technischen auch wesentliche organisatorische Herausforderungen, die zu beachten sind. Bevor die technische Seite der Umsetzung begonnen wird, sollte auf der organisatorischen Seite eine Unternehmensstrategie zur Überwachung und Störungserkennung festgelegt werden. Dies umfasst alle Teile der Organisation, insbesondere Entwicklung, Betrieb und Support. Bei der Befragung wurden wichtige strategische Aspekte für eine solche Unternehmensstrategie identifiziert: Unternehmensziele, Stellenwert, involvierte Partner, Zusammenarbeit und Verantwortlichkeiten und Umgang mit Insellösungen.

Unternehmensziele. Unternehmensziele in Bezug auf die Störungserkennung sind ein wesentlicher Treiber für die gesamte Umsetzung der Störungserkennung. Diese sollte von »ganz oben«, also vom Management festgelegt werden. Verantwortliche Personen sollten für die Umsetzung solcher Unternehmensziele bestimmt werden. Diese Ziele definieren, was das Unternehmen mit der Störungserkennung erreichen will und wie diese umgesetzt werden sollen. Beispiele für Ziele sind der Grad des Service-Levels oder der Kundenfokus.

Stellenwert. Zu der Unternehmensstrategie einer Störungserkennung gehört die Etablierung des Stellenwertes und der Wertschätzung der Störungserkennung im gesamten Unternehmen. Ist kein hoher Stellenwert vorhanden oder wird dieser von der Geschäftsführung nicht kommuniziert, wird es schwierig eine unternehmensübergreifende Störungserkennung zu implementieren. Störungserkennung muss schon bei der Entwicklung von Lösungen berücksichtigt werden, da es Auswirkungen auf Hardware- und Softwarelösungen gibt.

Involvierte Partner. Sind Partner und deren Systeme in Entwicklungs- oder Betriebsprozesse involviert, ist der richtige Umgang mit diesen Partnern entscheidend. Die Grundlage der Zusammenarbeit sollten klar definierte Service-Levels, Verpflichtungen und Kommunikationswege sein. Auf operativer Ebene sollte ein regelmäßiger Dialog geführt werden, sodass die Ansprechpartner sich kennen und nicht erst im Störungsfall miteinander kommunizieren. Dazu gehört auch, proaktiv über Dinge wie Software-Updates, Systemmigrationen, Wartungsintervalle, etc. zu kommunizieren. Schließlich sind involvierte Partner ein Teil der Gesamtlösung und sollten dementsprechend auch immer mit ins »Geschehen« eingegliedert werden.

Zusammenarbeit und Verantwortlichkeiten. Da Störungserkennung eine unternehmensweite Aufgabe darstellt, ist die Zusammenarbeit der verschiedenen involvierten Abteilungen wichtig. Eine gute Zusammenarbeit zeichnet sich durch eine offene Kommunikation und die Verfolgung gemeinsamer Ziele aus. Interessenskonflikte aufgrund von historischen Gegebenheiten, wie der bereits vorhandenen Insellösung oder den schon immer eingesetzten »alten« Standards, sollten ausgeräumt werden. Auch gehört zu einer guten Zusammenarbeit eine klare Verteilung der Verantwortlichkeiten.

Insellösungen. Insellösungen sind bestehende Lösungen einzelner Unternehmensbereiche oder Mitarbeiter, die zum Teil »historisch gewachsen« sind oder einmal als Provisorium gedacht waren. Der Umgang mit solchen Insellösungen sollte in der Unternehmensstrategie definiert werden. Auch wenn solche Lösungen oft nicht standardkonform sind, erfüllen sie im Alltag der Mitarbeiter eine wichtige Rolle, sodass bei einer Ablösung die Befürchtung besteht, ein notwendiges Werkzeug zu verlieren.
Hier ist im Einzelfall zu klären, ob eine Ablösung durch ein Standardsystem oder eine Integration sinnvoll ist. Allzwecklösungen zur Überwachung bieten zwar viele Funktionen, aber nicht den vollen Funktionsumfang einer spezialisierten Lösung. Besteht keine Option, solche Insellösungen durch andere standardisierte Lösungen zu ersetzen, ist es notwendig, diese in das Gesamtsystem zu integrieren und ggf. so anzupassen, damit die Integration einem gewissen Standard genügt.

7.2 Standardisierung

Standardisierung kann die Störungserkennung in verteilten IT-Systemen erheblich vereinfachen. Standardisierung stellt allerdings eine Herausforderung dar, da es technische, organisatorische und politische Hürden zu überwinden gilt. Standardisierung sollte deswegen mit in die Unternehmensstrategie aufgenommen werden. Unternehmensinterne Standards können Mehrwerte bieten, indem sie Mindestanforderungen für Qualität und Beobachtbarkeit stellen, die Integration von Systemen in die Störungserkennung vereinfachen und klare Anforderungen an die Entwicklung neuer Anforderungen definieren. Während Standards einen Mehraufwand für die Entwicklung bedeuten, bieten sie umgekehrt eine Garantie, dass die Anwendung im Betrieb in die Überwachung und Störungserkennung eingebunden werden kann.

Im Folgenden werden einige empfohlene Standards beschrieben, die für eine Störungserkennung in Betracht gezogen werden sollten:

Einheitliche Health Endpoints. Werden Health Endpoints für die Störungserkennung eingesetzt, sollten diese vereinheitlicht sein. Neben einem einheitlichen Datenformat sollten diese auch ein anwendungsunabhängiges Minimalset an Werten liefern, zum Beispiel Verfügbarkeit. Wichtig ist hier neben der Definition des Datenformats auch die Definition der Semantik, z. B. was es bedeutet, wenn eine Anwendung verfügbar ist. Ebenso wie das Datenformat sollte auch die Basis-Implementierung dieser Health Endpoints wiederverwendbar sein, um die Implementierung in weiteren Anwendungen zu vereinfachen. Einheitliche Health Points schaffen Vergleichbarkeit, vereinfachen die Integration und Konfiguration und verbessern das Verständnis der überwachten Werte.

Einheitliche Exception Policy. Eine einheitliche Exception Policy mit dem Einsatz von Fehlernummern erleichtert die Störungserkennung und -behebung. Sind Fehler nicht eindeutig zuordenbar, erschwert dies die Kommunikation mit dem Kunden und die Störungsbehebung. Die Verwendung von mehrstufigen Fehlernummern vereinfacht die Zuordnung des Fehlers auf Unternehmen, System und Komponente, wodurch der richtige Ansprechpartner identifiziert werden kann. Fehler von Dritten können mit einem entsprechenden Präfix in solch eine Struktur eingebettet werden, auch wenn der Drittanbieter selbst keine Fehlermeldung verwendet. Wichtig bei der Fehlerbehandlung ist, dass Fehlernummern auch korrekt zwischen den Systemen weitergegeben werden. In der Exception Policy sollte daher definiert werden, in welcher Form Fehler an wen zurückgegeben werden. Beim Aufruf einer API ist beispielsweise ein anderes Fehlerformat sinnvoll als in einer Benutzeroberfläche.

Logging Policy. Logging ist eine der meistverbreiteten Technologien im Bereich der Störungserkennung. Jedoch ist es entscheidend, wie es eingesetzt wird. Nur Loggen, damit »gelogged« ist, hat nur einen begrenzten Mehrwert. Vielmehr ist es wichtig, so zu loggen, dass das Log später zur Analyse von Störungen herangezogen werden kann. Auch ist es zu empfehlen, Logs an einem zentralen Ort zusammenzutragen. Weiterhin ist es vorteilhaft, Daten strukturiert in Logs zu schreiben (z. B. als JSON oder XML).

Richtlinien für Ende-zu-Ende-Tests. Ein Ende-zu-Ende-Test ist vergleichbar mit einem Integrationstest, bei dem beispielsweise ein Request von seinem Entstehungsort bis zu der entsprechenden Antwort getestet wird. So ist es möglich, die Funktionstüchtigkeit der involvierten Schnittstellen und der Hard- und Softwarekomponenten zu verifizieren. Ein Ende-zu-Ende-Test überwacht die Funktion aus Kundensicht. Bei der Implementierung empfiehlt es sich, in allen Komponenten einheitlich vorzugehen, was Parameter, Namen der Testnutzer, Resultate, etc. angeht.

7.3 Organisation

Störungserkennung ist eine organisatorische Herausforderung, was sich auch daran zeigt, dass viele befragte Experten organisatorische Herausforderungen in den Vordergrund stellten. Die wichtigste Best Practice ist offene Kommunikation und Zusammenarbeit zwischen Abteilungen. Dies betrifft sowohl die Entwicklung neuer Systeme als auch den Betrieb. Schon von Anfang an sollten alles Stakeholder integriert werden.

Klar definierte Prozesse helfen, Kommunikationsfehler zu verhindern und Abläufe zu verbessern. Dazu gehören Support-Prozesse wie Störungsmeldung und Eskalation sowie Entwicklungsprozesse wie Neuentwicklungen und Integration von Drittsystemen. Verantwortlichkeiten und Zuständigkeiten sind für solche Prozesse ebenfalls verbindlich festzuhalten. Definierte Verantwortlichkeiten für eine Störung von der Meldung bis zur Behebung, zum Beispiel durch ein Ticket-System, bieten wesentliche Vorteile.

Prozesse mit Dritten müssen ebenfalls definiert werden, insbesondere des Maßes der Verantwortung, die jeder Partner trägt. Wer für den Support eines Prozesses verantwortlich ist, sollte alle Abhängigkeiten dieses Prozesses zu Dritten sowie die entsprechenden Ansprechpartner kennen.

Ist eine Störung aufgetreten, spielt der Nachfolgeprozess eine wichtige Rolle. Dazu ist eindeutig zu beschreiben, wie eine Störung behoben werden soll, wem die Störung gemeldet wird und schlussendlich auch wer Maßnahmen zur Verbesserung der Überwachung und Prozesse festgelegt, falls notwendig.

7.4 Softwarewerkzeuge

Eine zentrale Software zur Überwachung und Störungserkennung ist notwendig, um den Überblick in komplexen, verteilten Systemlandschaften zu behalten. Damit verbunden sollten alle notwendigen Daten in einem System zusammengeführt werden, damit diese miteinander korreliert werden können.

Eigenentwicklung oder von der Stange? Viele Unternehmen stellen sich die Frage, ob eine selbst entwickelte Softwarelösung für die Störungserkennung geeigneter wäre als eine Softwarelösung vom Markt. Moderne Softwarelösungen bieten eine Vielzahl von Funktionen und Möglichkeiten für die Integration von Daten. Insbesondere Werkzeuge zum Application Performance Monitoring wurden spezifisch für die Anforderungen verteilter, dynamischer Systemlandschaften entwickelt. Eine Eigenentwicklung kann die Qualität und den Funktionsumfang solcher kommerziellen Lösungen nur schwer erreichen. Stattdessen ist es zielführender, bestehende Anwendungen beispielsweise mittels Plug-Ins um unternehmensspezifische Funktionen und Datenformate zu erweitern. So können Standardlösungen an spezifische Gegebenheiten (z. B. neue Anforderungen oder neue Technologien) angepasst werden. Bei der Auswahl ist auch zu beachten, dass nicht jede Software für jedes Unternehmen gleichermaßen geeignet ist. Umfangreiche Werkzeuge, die auf Großunternehmen ausgelegt sind, sind für kleine und mittelständische Unternehmen wahrscheinlich überdimensioniert. Ein Befragter fasste dies so zusammen: Kleine Unternehmen brauchen auch kleine Lösungen.

Konfiguration und Integration. Moderne Werkzeuge bieten umfangreiche Möglichkeiten zur Konfiguration von Systemen, Datenquellen, Regeln, Alarmen, etc. Obwohl moderne Lösungen versuchen, den Einrichtungsaufwand zu minimieren, sollte der Aufwand und die benötigte Expertise zur Konfiguration nicht unterschätzt werden. Nur mit fachkundiger Konfiguration lassen sich die umfangreichen Möglichkeiten der Werkzeuge ausschöpfen. Einige Befragte waren der Meinung, dass Integration und Konfiguration wichtiger sind als die Auswahl des Werkzeuges. Die Herausforderung sei, das, was möglich ist, auch zu tun.

Insellösungen. Insellösungen sind für spezielle Einsatzzwecke oft nur unter Qualitätsverlust und großem Aufwand in ein übergreifendes Überwachungswerkzeug zu überführen. Statt Lösungen zu ersetzen, sollte geprüft werden, ob diese integriert werden können. Die Schaffung neuer Insellösungen sollte aber nach Möglichkeit vermieden werden. Jedoch ist es wichtig, Kosten und Nutzen bei Ersetzung oder Integration einer Insellösung gegenüberzustellen. Vor allem sollte der Mehrwert für ein Ersetzen oder eine Integration klar definiert sein. Ein Weg, um Mehrwerte zu finden, ist, die »pain points« der bestehenden Lösung zu identifizieren.

7.5 Anwendungen

Anwendungen sind alle Softwarekomponenten eines verteilten IT-Systems, die für die Leistungserbringung mitverantwortlich sind. Gemeint ist hier also die »Produktivsoftware«, die einen Geschäftszweck erfüllt und nicht die Überwachungswerkzeuge.

Anwendungen müssen in die Überwachung und Störungserkennung integriert werden. Bei den Best Practices wird unterschieden, ob eine bestehende Anwendung angebunden werden soll oder ob eine neue Anwendung entwickelt wird, die die notwendigen Features zur Integration hat.

Bestehende Anwendungen. Bestehende Anwendungen sollten in den Überwachungsprozess zur Störungserkennung integriert werden. Dafür müssen technische Gegebenheiten erfüllt sein, die gerade Legacy-Anwendungen nicht haben. Aktuelle

Überwachungswerkzeuge bieten zahlreiche Möglichkeiten zur Integration, sodass es wahrscheinlich ist, eine Möglichkeit zu finden, die Integration zu bewerkstelligen. Dabei sind direkte Datenbindungen über APIs, Datenbanken etc. über indirekte Datenanbindung, beispielsweise durch das Auslesen von Log-Dateien zu bevorzugen. Diese bieten Vorteile, was die Aktualität, Präzision und den Integrationsaufwand angeht.

Muss eine Anwendung angepasst bzw. erweitert werden, müssen Kosten und Nutzen der Anpassung abgeschätzt werden und die Mehrwerte der Anpassung bezüglich Beobachtbarkeit herausgearbeitet werden. Welches Wissen fehlt für die Störungserkennung? Inwieweit hilft die Anpassung, dieses Wissen zu gewinnen?

Entwicklung neuer Anwendungen. Neue Anwendungen können entweder als Komplett-Anwendung gekauft oder aber auch eigens entwickelt werden. Die Einflussmöglichkeiten auf zugekaufte Software sind grundsätzlich geringer als auf Eigenentwicklungen. In beiden Fällen, sind Prozessverantwortliche von Anfang an miteinzubeziehen, damit die relevanten Anforderungen wie Fehlerfälle, Messwerte, Service-Level und Ziele schon zu Beginn berücksichtigt werden. Beim Einkauf von Software können diese in den Auswahlprozess einfließen, bei Eigenentwicklungen in die Spezifikation.

Während des Entwicklungsprozesses sollten Entwicklungsabteilung und Betrieb eng miteinander kommunizieren, sodass eine reibungslose Integration und Betriebsaufnahme erreicht werden können. In gleichem Maße sollten alle Verantwortlichen für verbundene Systeme miteinbezogen werden, sodass die Auswirkungen der neuen Anwendung auf das gesamte verteilte IT-System abgeschätzt werden können. Bei der Abnahme der Software sollte Beobachtbarkeit für die Störungserkennung Teil der Abnahmekriterien sein und genauso wie Funktionalität Vorbedingungen für die Inbetriebnahme sein.

Die Entwicklung sollte auf definierte unternehmensinterne Standards und, wenn vorhanden, Vorlagen zurückgreifen, insbesondere in Bezug auf Logging und Exception Policies. Sind keine unternehmensweiten Standards definiert, sollten die Entwickler eigene Best Practices definieren, mit der Aussicht, diese in folgenden Projekten wiederzuverwenden und so einen De-Facto-Standard zu etablieren, soweit dies eben möglich ist.

7.6 Systeme von Dritten

Die Anbindung von Systemen Dritter stellt besondere Herausforderungen dar, da deutlich weniger Möglichkeiten vorhanden sind, auf diese Systeme Einfluss zu nehmen. In den Befragungen wurden einige Best Practices identifiziert, um mit diesen Herausforderungen umzugehen.

Bestehende Drittsysteme. Um ein bestehendes Drittsystem zu überwachen, muss dieses erst einmal allen Stakeholdern bekannt sein. Verwendete Drittsysteme sollten dem Prozessverantwortlichen gemeldet werden und Betrieb und Support bekannt sein. So ist es möglich, im Falle einer Störung direkt mit dem Betreiber des Systems zu kommunizieren.

Für Drittsysteme sollte gemeinsam mit dem Prozessverantwortlichen ein Überwachungskonzept erstellt werden, ähnlich wie bei eigenen Systemen. Darin sind Abhängigkeiten, mögliche Fehler, Überwachungsmöglichkeiten, Ansprechpartner, etc. festzuhalten. Aufgrund der begrenzten Beobachtbarkeit von Drittsystemen kann es nötig sein, hierbei Kompromisse einzugehen. Im selben Zug sollte der Einfluss des Drittsystems auf das Gesamtsystem abgeschätzt werden, insbesondere in Bezug auf die Folgen eines Ausfalls.

Sind Service-Levels mit dem Betreiber des Drittsystems vereinbart, empfiehlt es sich, diese selbst an den Schnittstellen zu messen und sich nicht auf die Daten des Anbieters zu verlassen. So kann zum Beispiel ein System aus dem Netzwerk des Anbieters erreichbar sein, aus dem des Kunden aber nicht, was die Überwachung des Anbieters nicht ohne Weiteres erkennen kann.

Neue Drittsysteme. Werden Drittsysteme erst in ein bestehendes verteiltes IT-System integriert, ist bei der Beschaffung Spielraum für Verhandlungen von Service-Levels vorhanden. So kann die Auswahl auf Anbieter beschränken werden, die gewisse Mindestanforderungen erfüllen. Relevante Mindestanforderungen an die Störungserkennung betreffen Service-Levels, Verfügbarkeit des Supports, Meldepflichten, Beobachtbarkeit, Zugriff auf Rohdaten wie Logs. Auch hier muss, je nach Art der Anwendung und der eigenen Verhandlungsposition, auf Kompromisse eingegangen werden. Als Teil der Strategie kann es sinnvoll sein, unternehmenseigene Mindeststandards zu definieren, unter denen eine Drittanwendung nicht eingebunden werden darf. Auch hier sollten Prozessverantwortliche von Anfang an mit eingebunden werden. Bevor ein Drittsystem angeschafft wird, sollten alle Stakeholder eine Abnahme bezüglich Überwachung und Störungserkennung durchführen.

8 Fazit und Ausblick

Die Untersuchung des state-of-the-art und die Expertenbefragungen zeigen, dass sich in der Störungserkennung große Möglichkeiten und Herausforderungen gegenüberstehen. Mit Cloud-Infrastrukturen, mobilen Endgeräten, Software-as-a-Service, Virtualisierung und Microservices hat die Komplexität verteilter IT-Systeme erheblich zugenommen. Auch wenn diese Technologien bereits seit Jahren verfügbar sind, gibt es abseits der »Early Adopter« eine Vielzahl von Firmen, die diese Technologien gerade einführen oder das erste Mal erproben. Hier stoßen bestehende Ansätze zur Überwachung an ihre Grenzen. Der Bedarf an umfangreicheren Lösungen ist bereits bekannt, aber der Lösungsweg ist noch unklar.

Softwareanbieter und Berater haben auf diese Problemlage reagiert und bieten neue Lösungen an oder erweitern die bestehenden. Out-Of-The-Box-Lösungen gibt es allerdings nicht, es ist immer ein Anteil individueller Konfiguration und Implementierung notwendig. Auf unterster Ebene betrifft dies die im verteilten IT-System eingesetzte Software, an deren Messwerte erhoben und Störungen erkannt werden müssen. Wenn bestehende Anwendungen keine geeigneten Schnittstellen bieten, gibt es mittels moderner Überwachungswerkzeuge andere Möglichkeiten, an die Daten zu kommen, was aber auch einen Mehraufwand bedeutet. Die erfassten Daten müssen allerdings auch zusammengeführt und ausgewertet werden. Hier gibt es zwar erste Ansätze einer Automatisierung, aber was ein Normalfall und was ein Störungsfall ist, muss immer noch Großteils manuell definiert werden.

Die Hürde hier ist nicht die technische Machbarkeit, sondern die Verfügbarkeit von Ressourcen und Fachkenntnis. Da Störungserkennung nicht unmittelbar zur Wertschöpfung beiträgt, fehlt es manchmal am notwendigen Problembewusstsein und der notwendigen Wertschätzung im Unternehmen.

Deswegen sollte Störungserkennung nicht nur als technische, sondern als organisatorische Herausforderung verstanden werden. Die Befragten waren überwiegend der Meinung, dass die heutigen technischen Möglichkeiten ausreichend (wenn auch nicht perfekt) sind, sie aber nicht ausgeschöpft werden. Störungserkennung betrifft alle Unternehmensbereiche, oft kommunizieren Systeme verschiedener Abteilungen miteinander. Entwicklung, Fachabteilung und Support müssen zusammenarbeiten, gemeinsame Ziele haben und kommunizieren, damit der zuverlässige Betrieb gelingt. Dafür muss auch die Geschäftsführung hinter dem Thema Störungserkennung stehen und den Mitarbeitenden die notwendigen Freiräume im Tagesgeschäft schaffen.

Trotz dieser doppelten Herausforderung zeigen die vielen Positivbeispiele in den Befragungen, wie Entwicklung und Organisation gestaltet werden kann, um mit Überwachung nicht nur den Betrieb zuverlässiger machen zu können, sondern auch andere Mehrwerte generieren kann. Nebeneffekte einer ausgebauten Überwachung sind die Verfügbarkeit von Geschäftskennzahlen, verbesserte Zusammenarbeit, höhere Kundenzufriedenheit, geringere Aufwände für Entwicklung und Support und bessere Software.

Überwachung und Störungserkennung sind angesichts der Komplexität und des Stellenwerts heutiger IT-Systeme kein »Nice-To-Have« mehr, sondern kritischer Bestandteil der Unternehmensinfrastruktur. Auch wenn der Initialaufwand hoch ist, zeigten die Expertengespräche, dass investierter Aufwand sich langfristig wirtschaftlich lohnt.

Moderne Softwarelösungen bieten umfangreiche Möglichkeiten, diese Potenziale auch unter schwierigen Ausgangsbedingungen zu verwirklichen. Wie es ein Befragter formulierte: »Möglich ist sehr vieles, man muss es auch tun.«

9 Kurzvorstellung der interviewten Experten

Die Autoren der Studie möchten Ihren besonderen Dank den 29 interviewten Experten aussprechen. Die Experten hatten die Option, das Interview entweder anonym zu führen oder in der Studie erwähnt zu werden. Auf ihren Wunsch werden folgende Experten kurz vorgestellt:

Firma	**Camunda**
Name	*Jakob Freund*
Beruf	*CEO*
Webseite	*www.camunda.com*
Kontakt	*+49 30 664 04 09 00*

Firma	**Dynatrace**
Name	*Roman Spitzbart*
Beruf	*Senior Director Sales Engineering EMEA*
Webseite	www.dynatrace.com

Firma	**QMETHODS – Business & IT Consulting GmbH**
Name	*Matthias Scholze, Dipl. Inf.*
Beruf	*Enterprise IT Architekt & CEO*
Webseite	*www.qmethods.de*
Kontakt	*matthias.scholze@qmethods.com*

Firma	**Paessler AG**
Name	*Mathias Hengl*
Beruf	*Software Architect Product Development*
Webseite	*www.paessler.de*
Kontakt	*www.xing.com/profile/Mathias_Hengl*

Firma	**Elasticsearch**
Name	*Michaela Herzig*
Beruf	*Senior Marketing Manager*
Webseite	www.elastic.co
Kontakt	michaela.herzig@elastic.co

10 Über Fraunhofer

Fraunhofer-Gesellschaft

Die Fraunhofer-Gesellschaft mit Sitz in Deutschland ist die weltweit führende Organisation für anwendungsorientierte Forschung. Mit ihrer Fokussierung auf zukunftsrelevante Schlüsseltechnologien sowie auf die Verwertung der Ergebnisse in Wirtschaft und Industrie spielt sie eine zentrale Rolle im Innovationsprozess. Sie ist Wegweiser und Impulsgeber für innovative Entwicklungen und wissenschaftliche Exzellenz. Mit inspirierenden Ideen und nachhaltigen wissenschaftlich-technologischen Lösungen fördert die Fraunhofer-Gesellschaft Wissenschaft und Wirtschaft und wirkt mit an der Gestaltung unserer Gesellschaft und unserer Zukunft.

Interdisziplinäre Forschungsteams der Fraunhofer-Gesellschaft setzen gemeinsam mit Vertragspartnern aus Wirtschaft und öffentlicher Hand originäre Ideen in Innovationen um, koordinieren und realisieren systemrelevante, forschungspolitische Schlüsselprojekte und stärken mit werteorientierter Wertschöpfung die deutsche und europäische Wirtschaft. Internationale Kooperationen mit exzellenten Forschungspartnern und Unternehmen weltweit sorgen für einen direkten Austausch mit den einflussreichsten Wissenschafts- und Wirtschaftsräumen.

Die 1949 gegründete Organisation betreibt in Deutschland derzeit 74 Institute und Forschungseinrichtungen. Rund 28 000 Mitarbeiterinnen und Mitarbeiter, überwiegend mit natur- oder ingenieurwissenschaftlicher Ausbildung, erarbeiten das jährliche Forschungsvolumen von 2,8 Milliarden Euro. Davon fallen 2,3 Milliarden Euro auf den Leistungsbereich Vertragsforschung. Rund 70 Prozent davon erwirtschaftet Fraunhofer mit Aufträgen aus der Industrie und mit öffentlich finanzierten Forschungsprojekten. Rund 30 Prozent steuern Bund und Länder als Grundfinanzierung bei, damit die Institute schon heute Problemlösungen entwickeln können, die in einigen Jahren für Wirtschaft und Gesellschaft entscheidend wichtig werden.

Die Wirkung der angewandten Forschung geht weit über den direkten Nutzen für die Auftraggeber hinaus: Fraunhofer-Institute stärken die Leistungsfähigkeit der Unternehmen, verbessern die Akzeptanz moderner Technik in der Gesellschaft und sorgen für die Aus- und Weiterbildung des dringend benötigten wissenschaftlich-technischen Nachwuchses.

Hochmotivierte Mitarbeiterinnen und Mitarbeiter auf dem Stand der aktuellen Spitzenforschung stellen für uns als Wissenschaftsorganisation den wichtigsten Erfolgsfaktor dar. Fraunhofer bietet daher die Möglichkeit zum selbstständigen, gestaltenden und zugleich zielorientierten Arbeiten und somit zur fachlichen und persönlichen Entwicklung, die zu anspruchsvollen Positionen in den Instituten, an Hochschulen, in Wirtschaft und Gesellschaft befähigt. Studierenden eröffnen sich aufgrund der praxisnahen Ausbildung und des frühzeitigen Kontakts mit Auftraggebern hervorragende Einstiegs- und Entwicklungschancen in Unternehmen.

Namensgeber der als gemeinnützig anerkannten Fraunhofer-Gesellschaft ist der Münchner Gelehrte Joseph von Fraunhofer (1787–1826). Er war als Forscher, Erfinder und Unternehmer gleichermaßen erfolgreich.

Fraunhofer IAO

Grundlage der Arbeiten am Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO und am kooperierenden Institut für Arbeitswissenschaft und Technologiemanagement IAT an der Universität Stuttgart ist die Überzeugung, dass unternehmerischer Erfolg in Zeiten globalen Wettbewerbs vor allem bedeutet, neue technologische Potenziale nutzbringend einzusetzen. Deren erfolgreicher Einsatz wird vor allem durch die Fähigkeit bestimmt, kunden- und mitarbeiterorientiert Technologien schneller

als die Mitbewerber zu entwickeln und anzuwenden. Dabei müssen gleichzeitig innovative und anthropozentrische Konzepte der Arbeitsorganisation zum Einsatz kommen. Die systematische Gestaltung wird also erst durch die Bündelung von Management- und Technologiekompetenz ermöglicht. Die ganzheitliche Betrachtung bei der Projektbearbeitung gewährleistet, dass wirtschaftlicher Erfolg, Mitarbeiterinter-essen und gesellschaftliche Auswirkungen immer gleichwertig berücksichtigt werden.

Durch die enge Kooperation mit dem Institut für Arbeitswissenschaft und Technologiemanagement IAT der Universität Stuttgart verbindet das Fraunhofer IAO universitäre Grundlagenforschung, anwendungsorientierte Wissenschaft und wirtschaftliche Praxis.

Unter einer gemeinsamen Institutsleitung arbeiten am Fraunhofer IAO und dem IAT über 530 Mitarbeiter – vorwiegend Ingenieure, Informatiker, Wirtschafts- und Sozialwissenschaftler – interdisziplinär zusammen. Zur Bearbeitung der Forschungsaufträge stehen mehr als 14 200 m² moderner Büros, Labore und Demonstrationszentren zur Verfügung.

Forschungsprojekte werden in enger Zusammenarbeit mit der mittelständischen Industrie oder mit Großunternehmen im direkten Auftrag durchgeführt. Die Institute arbeiten in öffentlichen Forschungsprogrammen des Bundesministeriums für Bildung und Forschung (BMBF), Bundesministeriums für Wirtschaft und Energie (BMWi) und der Deutschen Forschungsgemeinschaft (DFG), in Programmen der Europäischen Union sowie regionalen Förderprogrammen der Landesregierung von Baden-Württemberg mit.

Forschungsbereich Digital Business

Die »richtige« IT ist eine wesentliche Grundlage für Erfolg und Wettbewerbsfähigkeit von Unternehmen. Besonders wichtig dabei: Sie muss sich intuitiv und komfortabel bedienen lassen. Nur dann werden die interaktiven Anwendungen in den Unternehmen akzeptiert und optimal genutzt. Voraussetzung dafür ist eine klare Analyse der Anforderungen eines Unternehmens hinsichtlich seiner Dienstleistungen und Produkte, seiner Kunden und Partner sowie seiner Organisation und der Mitarbeiter. Wir unterstützen Unternehmen und Institutionen dabei, mit Hilfe innovativer, aber auch bewährter Informationstechnik, ihre Aufgaben effizienter und kostengünstiger zu erfüllen.

Die Fokusthemen des Geschäftsfeldes Informations- und Kommunikationstechnik sind:

- Gestaltung abteilungs- und organisationsübergreifender Informationsprozesse (Stammdatenmanagement, Portale, mobile Dienste, IT-Sicherheit, elektronischer Geschäftsverkehr)
- Konzeption und Auswahl von Unternehmenssoftware (Dokumenten-, Geschäftsprozess-, Kundenbeziehungsmanagement, Produktinformationsmanagement und Enterprise Resource Planning)
- Design und Optimierung von IT-Architekturen (Web Services, Serviceorientierte Architekturen, Software as a Service, Cloud und Grid Computing)
- Entwicklung, Bewertung und Optimierung interaktiver und intelligenter IT-Systeme (Oberflächengestaltung, Usability, intelligente service-basierte Systeme, Backend-Integration und Einführungsstrategien)
- Definition und Verwendung von Standards und Schnittstellen (Electronic Business, Enterprise Application Integration, Stammdatenmanagement, Styleguides und Design Patterns Libraries)
- Beratungsdienstleistungen zu Überwachung und Störungserkennung in verteilten IT-Systemen
- Überwachung von kritischen Softwaresystemen (Konzeption, Beratung, eigene Software »Business Service Monitoring«)

Unsere Leistungen bauen auf fundierter Marktkenntnis und branchenübergreifenden Erfahrungen auf. Durch den Einsatz unserer praxiserprobten Methoden und die professionelle Durchführung der Projekte durch unsere Experten lässt sich der Unternehmenserfolg dauerhaft sichern. Über das Fraunhofer-Netzwerk können bei Bedarf weitere informationstechnologische oder organisatorische Kompetenzen hinzugezogen werden.

So ist gewährleistet, dass bei allen Umsetzungsprojekten aktuellstes technologisches Know-how zur Verfügung steht.

Business Innovation Engineering Center (BIEC)

Die vorliegende Studie ist Teil der Studienreihe »Digitale Transformation in KMU« des Business Innovation Engineering Centers (BIEC). Das BIEC hat als Entwicklungs- und Transferzentrum das Ziel, die digitalen Transformations- und Innovationsfähigkeit von kleinen und mittleren Unternehmen (KMU) in Baden-Württemberg zu steigern. Der Themenschwerpunkt »Data-driven Business und Künstliche Intelligenz« zielt darauf ab, insbesondere kleinen und mittleren Unternehmen die Potenziale Künstlicher Intelligenz (KI) näher zu bringen. Dabei werden u. a. folgende Fragestellungen behandelt:

- Wie können Geschäftsprozesse durch den Einsatz von KI optimiert werden?
- Wie unterstützt KI die Digitalisierung von Produkten und Dienstleistungen?
- Wie erfolgt der Umgang mit benötigten Daten?

Grundlage hierfür bilden umfangreiche Erfahrungen und Kompetenzen, bspw. aus den Bereichen KI-Engineering (KI-Einsatz und Nutzung von Data Science in Geschäftsprozessen), Extraktion und Klassifikation von Daten und Dokumenten (Potenziale durch Textverstehen und digitale Assistenten mit KI), Datenanalyse und KI für smarte Produkte und Dienstleistungen sowie KI-basiertes Service-Lifecycle-Management zur Identifikation von Potenzialen und zur Entscheidungsfindung. Im Rahmen von BIEC werden die spezifischen Chancen und Potenziale von KI identifiziert und ein systematischer Wissenstransfer durch den Einsatz unterschiedlichster Transferformate in die mittelständische Wirtschaft organisiert.

Für weitere Informationen besuchen Sie bitte unsere Website:

www.biec.iao.fraunhofer.de

11 Literaturverzeichnis

Anderson, E., Hoover, C., Li, X. and Tucek, J. (2009) 'Efficient tracing and performance analysis for large distributed systems', *2009 IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, pp. 1–10.

Arshad, F. (2014) *Failure characterization and error detection in distributed web applications,* Purdue University.

Beyer, B., ed. (2018) *The site reliability workbook: Practical ways to implement SRE* [Online], Sebastopol, CA, O'Reilly Media. Available at http://proquest.tech.safaribooksonline.de/9781492029496.

Bianco, P., Kotermanski, R. and Merson, P. (2007) *Evaluating a Service-Oriented Architecture,* Software Engineering Institute, Carnegie Mellon University CMU/SEI-2007-TR-015 [Online]. Available at http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8443.

BITKOM (2007) *Fehlerklassifikation für Software: Leitfaden* [Online]. Available at https://www.bitkom.org/sites/default/files/pdf/noindex/Publikationen/2008/Leitfaden/BITKOM-Leitfaden-Fehlerklassifikation-fuer-Software/080118-Fehlerklassifikation-fuer-Software-haftung.pdf.

Bitos, S. (2020) © *SergeyBitos* [Online]. Available at https://stock.adobe.com/de/images/informative-and-simple-dashboard-for-any-site-purposes-colorful-infographics-template-for-business-and-other-projects-admin-panel-interface-with-futuristic-blue-interface-vector-elements-set/287267888.

Blokdyk, G. (2018) *SOLID (object-oriented design) A Complete Guide*, 5STARCooks.

Bogner, A. and Menz, W. (2002) 'Das theoriegenerierende Experteninterview', in Bogner, A., Littig, B. and Menz, W. (eds) *Das Experteninterview: Theorie, Methode, Anwendung,* Wiesbaden, VS Verlag für Sozialwissenschaften, pp. 33–70.

Bruns, R. and Dunkel, J. (2010) *Event-driven architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse*, Springer-Verlag.

Desikan, S. and Ramesh, G. (2006) *Software testing: Principles and practices* [Online], Bangalore, India, Dorling Kindersley (India). Available at http://proquest.tech.safaribooksonline.de/9788177581218.

DPM Telecom (2019) *NMP OID: Introduction for Industry Professionals* [Online]. Available at https://www.dpstele.com/snmp/what-does-oid-network-elements.php (Accessed 21 October 2019).

Du, T. C., Li, E. Y. and Chang, A.-P. (2003) 'Mobile agents in distributed network management', *Communications of the ACM*, vol. 46, no. 7, pp. 127–132.

Dynatrace (2019) *Software Intelligence for the Enterprise Cloud* [Online]. Available at https://www.dynatrace.com/solutions/application-monitoring/ (Accessed 12 November 2019).

Echagüe, P. and Aijaz, A. (2018) *Managing Feature Flags*, O'Reilly Media, Inc.

Elasticsearch (2019) *Elasticsearch* [Online]. Available at https://www.elastic.co/de/blog/filebeat-modiles-access-logs-and-elasticsearch-storage-requirements (Accessed 2 May 2019).

Elektronik Kompendium (2019) *ISO/OSI-7-Schichtenmodell* [Online]. Available at https://www.elektronik-kompendium.de/sites/kom/0301201.htm (Accessed 29 October 2019).

EsperTech (ed) *Complex Event Processing - Products Data Sheet* [Online]. Available at http://static.espertech.com/EsperTech%20technical%20datasheet.pdf.

Ewaschuk, R. (2016) *Monitoring distributed systems* [Online]. Available at https://www.oreilly.com/ideas/monitoring-distributed-systems (Accessed 21 November 2018).

Freymann, A., Maier, F., Schaefer, K. and Boehnel, T. (2020) 'Tackling the Six Fundamental Challenges of Big Data in Research Projects by Utilizing a Scalable and Modular Architecture', in Wills, G., Kacsuk, P. and Chang Victor (eds) *IoTBDS 2020: 5th International Conference on Internet of Things, Big Data and Security,* SCITEPRESS – Science and Technology Publications, Lda, pp. 249–256.

Frühauf, K., Ludewig, J. and Sandmayr, H. (2002) *Software-Projektmanagement und-Qualitätssicherung*, vdf Hochschulverlag AG.

Gärtner, F. C. (1999) *Fehlererkennung und Fehlerdiagnose für verlässliche Systeme: Automatisierungstechnik vs. verteilte Systeme* [Online]. Available at https://faui1-files.cs.fau.de/filepool/publications/fehlererkennung-und-fehlerdiagnose-f-r-verl-ssliche-systeme-automatisierungstechnik-vs-verteilte-systeme.pdf.

Ghosh, S. (2014) *Distributed systems: an algorithmic approach*, Chapman and Hall/CRC.

Gopal, M. (1993) *Modern Control System Theory*, 2nd edn, New York, NY, USA, Halsted Press.

Helfferich, C. (2019) 'Leitfaden- und Experteninterviews', in Baur, N. and Blasius, J. (eds) *Handbuch Methoden der empirischen Sozialforschung,* Wiesbaden, Springer Fachmedien Wiesbaden, pp. 669–686.

Hitachi (2019) *Pentaho Platform* [Online]. Available at https://www.hitachivantara.com/en-us/products/data-management-analytics/pentaho-platform.html?source=pentaho-redirect (Accessed 11 November 2019).

IBM (2019) *IBM Streaming Analytics for IBM Cloud* [Online]. Available at https://www.ibm.com/cloud/streaming-analytics (Accessed 31 October 2019).

ISO/IEC (2010) *ISO/IEC 25010 System and software quality models*.

Kaplar, A., Vidaković, M., Luburić, N. and Ivanović, M. (2017) 'Improving a distributed agent-based Ant Colony Optimization for Solving Traveling Salesman Problem', *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1144–1148.

Kavulya, S. P., Joshi, K., Di Giandomenico, F. and Narasimhan, P. 'Failure Diagnosis of Complex Systems', in *Resilience Assessment and Evaluation of Computing Systems*, pp. 239–261.

Koetter, F., Kochanowski, M., Kintz, M., Kersjes, B., Bogicevic, I. and Wagner, S., eds. (2019) *Assessing Software Quality of Agile Student Projects by Data-mining Software Repositories*, SciTePress.

Kötter, F., Kochanowski, M. and Renner, T. (2014) *Überwachung in Geschäftsprozessen: Business Process Management Tools 2014 ; [Schwerpunktstudie zum Thema Überwachung für Geschäftsprozessmanagement-Werkzeuge* [Online], Stuttgart, Fraunhofer-Verl. Available at http://publica.fraunhofer.de/dokumente/N-307160.html.

Kufel, Ł. (2016) *Tools For Distributed Systems Monitoring* [Online], 41st edn, Poznan, Poland. Available at https://www.researchgate.net/publication/311863266_Tools_for_Distributed_Systems_Monitoring/fulltext/585e5e8908ae329d61f8dc24/311863266_Tools_for_Distributed_Systems_Monitoring.pdf.

Lehmann, K. and Freymann, A. (2018) 'Demo Abstract: Smart Urban Services Platform a Flexible Solution for Smart Cities', *ACM/IEEE International Conference on Internet of Things Design and Implementation: IoTDI 2018 : 17-20 April 2018, Orlando, Florida, USA : proceedings.* Orlando, FL, 4/17/2018 - 4/20/2018. Piscataway, NJ, IEEE, pp. 306–307.

Lethbridge, T. C., Sim, S. E. and Singer, J. (2005) 'Studying Software Engineers: Data Collection Techniques for Software Field Studies', *Empirical Software Engineering*, vol. 10, no. 3, pp. 311–341.

Loosen, W. (2016) 'Das Leitfadeninterview - eine unterschätzte Methode', in Averbeck-Lietz, S. and Meyen, M. (eds) *Handbuch nicht standardisierte Methoden in der Kommunikationswissenschaft,* Wiesbaden, Springer Fachmedien Wiesbaden, pp. 139–155.

Mace, J., Roelke, R. and Fonseca, R. (2016) 'Pivot Tracing: Dynamic Causal Monitoring for Distributed Systems', *2016 USENIX Annual Technical Conference (USENIX ATC 16).* Denver, CO, USENIX Association.

Mattern, F. (2005) *Verteilte Systeme - Fehlerursachen* [Online]. Available at https://www.vs.inf.ethz.ch/edu/WS0405/VS/Vorl.VertSys04_05-4.pdf.

McConnell, S. (2009) *Code Complete* [Online], 2nd edn, Sebastopol, O'Reilly Media Inc. Available at http://gbv.eblib.com/patron/FullRecord.aspx?p=488632.

Niedermaier, S., Koetter, F., Freymann, A. and Wagner, S. (2019) *On Observability and Monitoring of Distributed Systems: An Industry Interview Study* [Online]. Available at http://arxiv.org/pdf/1907.12240v1.

Oracle (2019a) *Oracle Business Activity Monitoring* [Online]. Available at https://www.oracle.com/middleware/technologies/business-activity-monitoring.html (Accessed 12 November 2019).

Oracle (2019b) *Oracle Stream Analytics* [Online]. Available at https://docs.oracle.com/en/middleware/fusion-middleware/osa/18.1/understanding-stream-analytics/overview-oracle-streaming-analytics.html (Accessed 18 May 2020).

Osherove, R. (2014) *The art of unit testing: With examples in C#* [Online], 2nd edn, Shelter Island, NY, Manning Publications. Available at http://proquest.tech.safaribooksonline.de/9781617290893.

Rajesh, R. V. (2017) *Spring 5.0 microservices: Build scalable microservices with Reactive Streams, Spring Boot, Docker, and Mesos* [Online], Birmingham, UK, Packt Publishing. Available at http://proquest.tech.safaribooksonline.de/9781787127685.

Richards, G., Yeoh, W., Chong, A. Y. L. and Popovič, A. (2019) 'Business intelligence effectiveness and corporate performance management: an empirical analysis', *Journal of Computer Information Systems*, vol. 59, no. 2, pp. 188–196.

Setter, M. (2017) *The Top Four Exception Tracking Services* [Online]. Available at https://blog.codeship.com/the-top-four-exception-tracking-services/? (Accessed 7 November 2019).

Siroker, D. and Koomen, P. (2013) *A/B testing: The most powerful way to turn clicks into customers*, Hoboken, NJ, Wiley.

Tarlinder, A. (2017) *Developer testing: Building quality into software* [Online], Boston, Addison-Wesley. Available at http://proquest.tech.safaribooksonline.de/9780134291109.

Vidačković, K., Renner, T. and Rex, S. (2010) *Marktübersicht Real-Time Monitoring Software: Event Processing Tools im Überblick ; [Forschungsprojekt] iC RFID - intelligent catering* [Online], Stuttgart, Fraunhofer Verl. Available at http://publica.fraunhofer.de/dokumente/N-143728.html.

Zawawy, H., Kontogiannis, K. and Mylopoulos, J. (2010) *Log Filtering and Interpretation for Root Cause Analysis* [Online], Romania. Available at https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5609556.

Zobel, D. and Behnsen, M. (2019) *SNMP, MIB, and OIDs—an Overview* [Online]. Available at https://kb.paessler.com/en/topic/653-how-do-snmp-mibs-and-oids-work (Accessed 20 October 2019).

Zvara, Z., G.N. Szabó, P., Hermann, G. and Benczúr, A. (2017) 'Tracing Distributed Data Stream Processing Systems', *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pp. 235–242.

BUSINESS INNOVATION ENGINEERING CENTER (BIEC)

Die vorliegende Studie ist Teil der Studienreihe »Digitale Transformation in KMU« des Business Innovation Engineering Centers (BIEC). Das BIEC hat als Entwicklungs- und Transferzentrum das Ziel, die digitalen Transformations- und Innovationsfähigkeit von kleinen und mittleren Unternehmen (KMU) in Baden-Württemberg zu steigern. Der Themenschwerpunkt »Data-driven Business und Künstliche Intelligenz« zielt darauf ab, insbesondere kleinen und mittleren Unternehmen die Potenziale Künstlicher Intelligenz (KI) näher zu bringen.

Die Bedeutung von vernetzten IT-Systemen ist in den letzten Jahren im Zuge der digitalen Transformation durch Technologien wie Cloud, Künstliche Intelligenz und dem Internet der Dinge stark gestiegen. Ungeachtet ihrer zahlreichen Vorteile weisen vernetzte IT-Systeme eine hohe Komplexität auf und erfüllen zunehmend Kernfunktionen von Unternehmen und Produkten. Ein sicherer und störungsfreier Betrieb ist daher unabdingbar.

Die vorliegende Marktstudie bietet einen Überblick über den Stand der Technik und Praxis für die Überwachung und Störungserkennung in vernetzten Systemen. Neben einer Marktuntersuchung wurden dafür 28 Experten befragt. Als Ergebnis zeigt die Studie Handlungsoptionen auf, stellt aktuell am Markt verfügbare technische Lösungen systematisch vor und beschreibt praktische Erfahrungen aus der Anbieter- und der Anwenderperspektive.

Für weitere Informationen besuchen Sie bitte unsere Website: www.biec.iao.fraunhofer.de