4.2 Ursachenanalyse (Root Cause Analysis – RCA)

> **Kurz erklärt**
>
> Ursachenanalyse bedeutet, systematisch zu untersuchen, warum ein Incident oder Problem entstanden ist.
>
> Ziel ist nicht, möglichst schnell einen Schuldigen zu finden.
>
> Ziel ist, die tatsächlichen technischen, organisatorischen oder prozessualen Ursachen zu verstehen, damit Wiederholungen verhindert oder Auswirkungen reduziert werden können.

Warum Ursachenanalyse wichtig ist

Ohne Ursachenanalyse werden Incidents häufig nur kurzfristig behoben.

Beispiele:

- Dienst wird immer wieder neu gestartet.
- Speicher wird regelmäßig manuell bereinigt.
- Benutzerkonten werden mehrfach entsperrt.
- VPN-Client wird immer wieder neu installiert.
- Druckwarteschlange wird regelmäßig geleert.
- Zertifikate werden erst nach Ablauf erneuert.

Solche Maßnahmen können kurzfristig helfen.

Sie beantworten aber nicht die Frage:

> Warum tritt der Fehler immer wieder auf?

Ursachenanalyse hilft dabei:

- wiederkehrende Störungen zu vermeiden,
- Risiken sichtbar zu machen,
- dauerhafte Lösungen vorzubereiten,
- Workarounds gezielt zu dokumentieren,
- Verbesserungen abzuleiten,
- und technische sowie organisatorische Schwachstellen zu erkennen.

Incident-Lösung und Ursachenanalyse unterscheiden

Incident Management	Problem Management / RCA
Service schnell wiederherstellen	Ursache verstehen
kurzfristiger Fokus	nachhaltiger Fokus
Workaround kann genügen	dauerhafte Lösung wird vorbereitet
Ziel: Benutzer wieder arbeitsfähig machen	Ziel: Wiederholung verhindern oder Risiko senken
Zeitdruck häufig hoch	Analyse kann länger dauern

Beispiel:

Ein Webservice ist nicht erreichbar.

Incident Management

- Dienststatus prüfen
- Dienst kontrolliert neu starten
- Erreichbarkeit testen
- Benutzer informieren

Ursachenanalyse

- Warum ist der Dienst abgestürzt?
- Warum wurde es nicht früher erkannt?
- Warum gab es keinen automatischen Neustart?
- Warum war kein Monitoring vorhanden?
- Warum trat der Fehler nach einem bestimmten Deployment auf?

> **Merke**
>
> Eine schnelle Wiederherstellung ist wichtig.
>
> Sie ersetzt aber keine Ursachenanalyse, wenn der Fehler wiederkehren kann.

Symptom, Ursache und Grundursache unterscheiden

Begriff	Bedeutung	Beispiel
Symptom	sichtbares Fehlverhalten	Benutzer kann sich nicht anmelden
direkte Ursache	unmittelbarer technischer Auslöser	Zertifikat ist abgelaufen

Begriff	Bedeutung	Beispiel
Grundursache	tiefer liegende Ursache	kein Prozess zur Zertifikatsüberwachung
beitragender Faktor	Umstand, der den Fehler begünstigt	Warnung wurde nicht an das richtige Team gesendet
Auswirkung	Folge für Benutzer oder Geschäft	Fachanwendung nicht nutzbar

Ein Problem besitzt häufig nicht nur eine Ursache.

Oft wirken mehrere Faktoren zusammen.

Beispiel:

Ein Server fällt aus, weil der Speicher voll ist.

Mögliche Ursachenebenen:

- Logdateien wachsen stark.
- Logrotation funktioniert nicht.
- Monitoring warnt zu spät.
- Alarm wird an eine nicht mehr gepflegte Verteilerliste gesendet.
- Verantwortlichkeit für den Server ist unklar.

Nur die Aussage „Speicher voll“ beschreibt noch nicht die vollständige Ursache.

Ursachenanalyse ist keine Schuldzuweisung

Ursachenanalyse soll nicht klären:

> “ Wer hat den Fehler gemacht?

Sondern:

> “ Welche Bedingungen haben dazu geführt, dass dieser Fehler entstehen oder unentdeckt bleiben konnte?

Eine schuldorientierte Kultur führt häufig dazu, dass:

- Informationen zurückgehalten werden,
- Fehler verschwiegen werden,
- Symptome beschönigt werden,

- Mitarbeitende defensiv reagieren,
- und echte Ursachen verborgen bleiben.

Eine lernorientierte Analyse fragt stattdessen:

- Welche Annahmen waren falsch?
- Welche Kontrolle hat gefehlt?
- Welche Information war nicht verfügbar?
- Welcher Prozess war unklar?
- Welche technische Schutzmaßnahme hätte geholfen?
- Welche Dokumentation war unvollständig?
- Welche Automatisierung hätte den Fehler verhindert?

Grundprinzip einer guten Ursachenanalyse

Eine gute Ursachenanalyse ist:

- faktenbasiert,
- nachvollziehbar,
- strukturiert,
- frei von vorschnellen Schuldzuweisungen,
- offen für mehrere Ursachen,
- mit Daten belegbar,
- und auf Verbesserungen ausgerichtet.

Sie trennt:

- Beobachtungen,
- Vermutungen,
- Hypothesen,
- bestätigte Fakten,
- Schlussfolgerungen,
- und Maßnahmen.

Beobachtung und Hypothese trennen

Ungeeignet:

> Die Firewall war schuld.

Besser:

>

> Verbindungen zur Anwendung auf Port 443 schlugen vom Standort Süd fehl. Eine Firewall-Regel ist eine mögliche Ursache und wird geprüft.

Ungeeignet:

> Der Benutzer hat etwas falsch gemacht.

Besser:

> Der Fehler trat nach einer Änderung der Berechtigungsgruppe auf. Die Auswirkungen der Gruppenänderung werden geprüft.

Ungeeignet:

> Das Update hat alles kaputt gemacht.

Besser:

> Der Incident begann 20 Minuten nach dem Update. Ein Zusammenhang ist möglich, aber noch nicht bestätigt.

> **Merke**
>
> Zeitlicher Zusammenhang ist ein Hinweis.
>
> Er ist noch kein Beweis.

Typischer Ablauf einer Ursachenanalyse

Ein möglicher Ablauf:

```
Problem beschreiben
        ↓
```

```
Fakten sammeln
      ↓
Zeitlinie erstellen
      ↓
Umfang und Auswirkungen bestimmen
      ↓
Hypothesen bilden
      ↓
Hypothesen prüfen
      ↓
Ursache und beitragende Faktoren bestimmen
      ↓
Workaround dokumentieren
      ↓
dauerhafte Lösung oder Verbesserungen ableiten
      ↓
Maßnahmen priorisieren und verfolgen
```

Nicht jede Organisation nutzt exakt diesen Ablauf.

Wichtig ist, dass die Analyse nachvollziehbar bleibt.

Problem klar beschreiben

Am Anfang sollte das Problem eindeutig beschrieben werden.

Eine gute Problembeschreibung enthält:

- betroffenen Service,
- beobachtetes Symptom,
- Zeitraum,
- betroffene Benutzer oder Standorte,
- Auswirkung,
- Häufigkeit,
- bekannte Auslöser,
- und aktuellen Workaround.

Ungeeignet:

> “ VPN macht Probleme.

Besser:

> Seit dem Client-Update vom 12.08.2026 verlieren mehrere Benutzer nach dem Ruhezustand die VPN-Verbindung. Betroffen sind Windows-Notebooks mit Client-Version 5.8. Der Workaround ist ein vollständiger Neustart des VPN-Clients.

Fakten sammeln

Geeignete Informationsquellen:

- Incident Records,
- Problem Records,
- Change Records,
- Monitoringdaten,
- Logdateien,
- Fehlermeldungen,
- Zeitstempel,
- Benutzerfeedback,
- Konfigurationsdaten,
- CMDB-Informationen,
- Herstellerhinweise,
- Netzwerkdaten,
- Backup- und Restore-Informationen,
- Sicherheitsmeldungen,
- Service-Statusmeldungen.

Wichtig ist, die Daten nicht nur zu sammeln, sondern sie fachlich einzuordnen.

Nicht jede Logmeldung ist relevant.

Nicht jeder Benutzerbericht beschreibt die technische Ursache.

Zeitlinie erstellen

Eine Zeitlinie hilft, Ereignisse in die richtige Reihenfolge zu bringen.

Beispiel:

Zeitpunkt	Ereignis
08:00 Uhr	Deployment abgeschlossen
08:20 Uhr	erste Fehlermeldung im Monitoring
08:25 Uhr	erste Benutzermeldung

Zeitpunkt	Ereignis
08:35 Uhr	Incident als P1 bewertet
08:45 Uhr	Rollback vorbereitet
09:05 Uhr	Workaround aktiv
09:30 Uhr	Service stabil

Eine Zeitlinie hilft bei Fragen wie:

- Was geschah vor dem Incident?
- Wann begann die Auswirkung?
- Welche Änderung war zeitlich relevant?
- Wann wurde der Incident erkannt?
- Wann wurde eskaliert?
- Welche Maßnahme hatte welchen Effekt?
- Wo entstanden Verzögerungen?

Umfang bestimmen

Zu prüfen ist:

- Sind einzelne Benutzer betroffen?
- Ist ein Team betroffen?
- Ist ein Standort betroffen?
- Sind mehrere Standorte betroffen?
- Sind externe Kunden betroffen?
- Sind alle Funktionen betroffen oder nur einzelne?
- Sind bestimmte Versionen betroffen?
- Sind bestimmte Geräte betroffen?
- Sind bestimmte Netzwerkpfade betroffen?
- Sind nur neue Sitzungen betroffen oder auch bestehende?

Der Umfang hilft, mögliche Ursachen einzugrenzen.

Beispiel:

Wenn nur ein Standort betroffen ist, sind zentrale Anwendung und Benutzerkonto möglicherweise weniger wahrscheinlich als Standortnetz, Routing, DNS oder lokale Firewall.

Änderungen prüfen

Viele Probleme stehen im Zusammenhang mit Änderungen.

Zu prüfen sind:

- Softwareupdates,
- Konfigurationsänderungen,
- Firewall-Regeln,
- DNS-Änderungen,
- Zertifikatswechsel,
- Gruppenrichtlinien,
- neue Berechtigungen,
- neue Versionen,
- Datenbankänderungen,
- Netzwerkumbauten,
- Lieferantenänderungen,
- Cloud-Konfigurationsänderungen.

Wichtig:

Ein Change ist nicht automatisch die Ursache.

Er ist zunächst ein relevanter Untersuchungsgegenstand.

5-Why-Methode

Die **5-Why-Methode** fragt mehrfach „Warum?“, um von einem Symptom zu tiefer liegenden Ursachen zu gelangen.

Beispiel:

Problem

Ein Webservice war nicht erreichbar.

Warum 1

Warum war der Webservice nicht erreichbar?

> Der Dienst war abgestürzt.

Warum 2

Warum ist der Dienst abgestürzt?

> Der Speicher war vollständig belegt.

Warum 3

Warum war der Speicher vollständig belegt?

> Logdateien sind stark angewachsen.

Warum 4

Warum wurden Logdateien nicht bereinigt?

> Die Logrotation war fehlerhaft konfiguriert.

Warum 5

Warum wurde die fehlerhafte Logrotation nicht erkannt?

> Es gab kein Monitoring für Logwachstum und keine regelmäßige Prüfung.

Mögliche Maßnahmen:

- Logrotation korrigieren,
- Monitoring ergänzen,
- Alarmgrenzen definieren,
- Runbook aktualisieren,
- Verantwortlichkeit festlegen.

Grenzen der 5-Why-Methode

Die 5-Why-Methode ist einfach und nützlich.

Sie hat aber Grenzen.

Risiken:

- zu lineares Denken,
- nur eine Ursache wird betrachtet,
- komplexe Zusammenhänge werden vereinfacht,
- Antworten hängen stark von der Perspektive ab,
- Schuldzuweisungen können entstehen,
- organisatorische Faktoren werden übersehen.

Deshalb sollte sie bei komplexen Incidents mit anderen Methoden kombiniert werden.

Ishikawa-Diagramm

Das **Ishikawa-Diagramm** wird auch Ursache-Wirkungs-Diagramm oder Fischgräten-Diagramm genannt.

Es hilft, mögliche Ursachen nach Kategorien zu sortieren.

Mögliche Kategorien in der IT:

- Mensch,
- Prozess,
- Technik,
- Umgebung,
- Organisation,
- Lieferant,
- Dokumentation,
- Daten,
- Sicherheit.

Beispiel:

Problem:

> Wiederkehrende VPN-Abbrüche.

Mögliche Ursachen:

Kategorie	Beispiele
Mensch	Benutzer versetzen Notebook in Ruhezustand
Prozess	Clientupdates werden nicht getestet
Technik	fehlerhafte VPN-Client-Version
Umgebung	instabile WLAN-Verbindung
Organisation	kein Owner für Clientstandard
Lieferant	bekannter Fehler in Version
Dokumentation	Workaround fehlt in Knowledge Base
Sicherheit	Zertifikatsprüfung schlägt fehl

Das Diagramm zwingt dazu, nicht nur eine technische Ursache zu betrachten.

Pareto-Analyse

Die Pareto-Analyse hilft, die wichtigsten Ursachen oder Kategorien zu erkennen.

Grundidee:

Ein kleiner Teil der Ursachen erzeugt häufig einen großen Teil der Auswirkungen.

Beispiel:

Kategorie	Anzahl Incidents pro Monat
Passwort und MFA	120
VPN	75
Drucker	40
Softwareinstallation	25
Sonstiges	20

Wenn Passwort/MFA und VPN den größten Anteil ausmachen, können Verbesserungen dort besonders viel Wirkung entfalten.

Die Pareto-Analyse ersetzt keine Ursachenanalyse.

Sie hilft bei der Priorisierung.

Kepner-Tregoe-Ansatz

Ein strukturierter Ansatz kann sein, zu unterscheiden:

- Was ist betroffen?
- Was ist nicht betroffen?
- Wo tritt es auf?
- Wo tritt es nicht auf?
- Seit wann tritt es auf?
- Seit wann nicht?
- Wie stark ist die Auswirkung?
- Welche Veränderung passt dazu?

Beispiel:

Frage	Antwort
Was ist betroffen?	VPN-Verbindung nach Ruhezustand
Was ist nicht betroffen?	VPN nach Neustart

Frage	Antwort
Wo tritt es auf?	Windows-Notebooks
Wo nicht?	macOS-Geräte
Seit wann?	seit Client-Version 5.8
Veränderung	Clientupdate am Vortag

Diese Gegenüberstellung hilft, Hypothesen gezielt einzugrenzen.

Fehlerbaum und Abhängigkeiten

Bei komplexen Services kann eine Abhängigkeitsanalyse helfen.

Zu prüfen sind:

- Anwendungen,
- Datenbanken,
- Identitätsdienste,
- Netzwerkverbindungen,
- DNS,
- Zertifikate,
- Firewalls,
- Load Balancer,
- Storage,
- Cloud-Dienste,
- externe APIs,
- Lieferantenservices.

Beispiel:

Eine Anwendung ist nicht erreichbar.

Mögliche Abhängigkeiten:

```
Benutzer
   ↓
Netzwerk
   ↓
DNS
   ↓
Load Balancer
   ↓
Webserver
```

```
   ↓
Anwendung
   ↓
Datenbank
   ↓
Storage
```

Wenn mehrere Anwendungen betroffen sind, kann eine gemeinsame Abhängigkeit wahrscheinlicher sein.

Datenqualität beachten

Ursachenanalyse ist nur so gut wie die verfügbaren Daten.

Probleme entstehen durch:

- fehlende Logs,
- falsche Zeitstempel,
- unterschiedliche Zeitzonen,
- unvollständige Tickets,
- veraltete CMDB,
- fehlende Change-Dokumentation,
- unklare Serviceabhängigkeiten,
- fehlende Monitoringdaten,
- nicht dokumentierte manuelle Eingriffe.

Eine wichtige Verbesserung kann daher sein:

> Die Datenbasis für zukünftige Analysen verbessern.

Korrelation und Kausalität unterscheiden

Nur weil zwei Ereignisse zeitlich zusammen auftreten, bedeutet das nicht, dass eines das andere verursacht hat.

Beispiel:

Ein Incident beginnt kurz nach einem Windows-Update.

Mögliche Schlussfolgerungen:

- Das Update ist die Ursache.

- Das Update hat nur eine bestehende Schwäche sichtbar gemacht.
- Ein anderer Change trat gleichzeitig auf.
- Der Incident wäre unabhängig davon eingetreten.

Deshalb müssen Hypothesen geprüft werden.

Hypothesen testen

Eine Hypothese sollte überprüfbar sein.

Beispiel:

Hypothese

VPN-Abbrüche entstehen durch Client-Version 5.8.

Test

- betroffene Geräteversionen vergleichen,
- nicht betroffene Geräte prüfen,
- bekannte Herstellerhinweise suchen,
- Testgerät mit älterer Version prüfen,
- kontrolliertes Update durchführen,
- Logs auswerten.

Ergebnis

Wenn nur Version 5.8 betroffen ist und ein Downgrade den Fehler beseitigt, wird die Hypothese stärker.

Trotzdem muss geprüft werden, ob weitere Faktoren beteiligt sind.

Kontrollierte Tests

Tests sollten möglichst kontrolliert durchgeführt werden.

Wichtig:

- nicht mehrere Änderungen gleichzeitig,
- Testumgebung nutzen, wenn möglich,
- Risiko bewerten,
- Messwerte vorher und nachher vergleichen,
- Ergebnisse dokumentieren,
- Rollback planen,
- Change-Vorgaben beachten.

Ungeeignet:

> Wir ändern DNS, Firewall, Zertifikat und Clientversion gleichzeitig.

Besser:

> Wir testen zuerst die Clientversion auf einem betroffenen Gerät und dokumentieren das Ergebnis.

RCA bei Major Incidents

Nach Major Incidents ist eine Ursachenanalyse besonders wichtig.

Zu betrachten sind nicht nur technische Ursachen.

Auch zu prüfen:

- Wurde der Incident rechtzeitig erkannt?
- Waren die Alarme verständlich?
- Wurde richtig priorisiert?
- War Ownership klar?
- Wurde rechtzeitig eskaliert?
- War Kommunikation ausreichend?
- Gab es einen Workaround?
- Waren Abhängigkeiten bekannt?
- Waren Lieferanten erreichbar?
- Wurde der Service stabil wiederhergestellt?

Ein Major Incident Review sollte nicht nur fragen:

> Warum ist das System ausgefallen?

Sondern auch:

> Warum waren die Auswirkungen so groß?

Technische Ursachen

Mögliche technische Ursachen:

- Softwarefehler,
- Hardwaredefekt,
- fehlerhafte Konfiguration,
- abgelaufenes Zertifikat,
- Netzwerkausfall,
- DNS-Fehler,
- Speicher voll,
- Datenbankproblem,
- fehlgeschlagenes Backup,
- Überlastung,
- falsche Berechtigungen,
- inkompatible Versionen.

Technische Ursachen sind oft sichtbar.

Sie sind aber nicht immer die tiefste Ursache.

Prozessuale Ursachen

Mögliche prozessuale Ursachen:

- Change wurde nicht ausreichend geprüft,
- Monitoring wurde nicht aktualisiert,
- Dokumentation war veraltet,
- Genehmigungsweg war unklar,
- Eskalationsweg fehlte,
- Runbook war unvollständig,
- Backup-Wiederherstellung wurde nie getestet,
- Serviceübergabe war unvollständig,
- Verantwortlichkeit war nicht festgelegt.

Prozessuale Ursachen erklären häufig, warum ein technischer Fehler nicht verhindert oder früher erkannt wurde.

Organisatorische Ursachen

Mögliche organisatorische Ursachen:

- unklare Ownership,
- fehlende Ressourcen,
- fehlende Schulung,
- unklare Prioritäten,
- fehlende Abstimmung zwischen Teams,

- ungeeignete Lieferantensteuerung,
- fehlende Vertretung,
- Silodenken,
- unklare Entscheidungskompetenz.

Diese Ursachen sind oft unangenehm, aber wichtig.

Wenn sie ignoriert werden, treten ähnliche Probleme wieder auf.

Menschliche Faktoren

Menschen machen Fehler.

Gute Ursachenanalyse fragt deshalb nicht nur:

> Warum hat jemand falsch gehandelt?

Sondern:

> Warum war dieser Fehler möglich oder wahrscheinlich?

Zu prüfen sind:

- unklare Oberfläche,
- fehlende Warnung,
- Zeitdruck,
- unvollständige Anleitung,
- fehlende Schulung,
- zu viele manuelle Schritte,
- ungünstige Berechtigungen,
- fehlende Prüfung,
- widersprüchliche Informationen.

Ziel ist, Systeme und Prozesse robuster zu machen.

Beitragende Faktoren dokumentieren

Nicht jede Ursache ist allein verantwortlich.

Beispiel:

Ein Datenbankserver fällt aus.

Beitragende Faktoren:

- Speicherplatz war knapp.
- Monitoringgrenze war zu hoch gesetzt.
- Alarm ging an falschen Verteiler.
- Bereinigungsskript war deaktiviert.
- Dokumentation war veraltet.
- Bereitschaft war nicht informiert.

Die Kombination führte zur Störung.

Deshalb sollten mehrere Faktoren dokumentiert werden.

Maßnahmen ableiten

Aus der Ursachenanalyse sollten konkrete Maßnahmen entstehen.

Gute Maßnahmen sind:

- eindeutig beschrieben,
- einer verantwortlichen Person oder Gruppe zugewiesen,
- terminiert,
- risikobewertet,
- überprüfbar,
- und nachverfolgbar.

Beispiele:

Ursache	Maßnahme
Zertifikatsablauf wurde nicht überwacht	Monitoring für Zertifikatsgültigkeit einführen
Logrotation war fehlerhaft	Konfiguration korrigieren und Test ergänzen
Workaround war unbekannt	Knowledge-Artikel erstellen
Eskalation erfolgte zu spät	Eskalationskriterien überarbeiten
Serviceabhängigkeit war unbekannt	CMDB-Beziehung ergänzen
Lieferant reagierte zu langsam	Lieferanteneskalationsweg prüfen

---

Maßnahmen priorisieren

Nicht jede Maßnahme kann sofort umgesetzt werden.

Zu bewerten sind:

- Nutzen,
- Aufwand,
- Risiko,
- Kosten,
- Sicherheitsrelevanz,
- Häufigkeit des Problems,
- Kritikalität des betroffenen Service,
- Verfügbarkeit eines Workarounds,
- Abhängigkeit von Lieferanten,
- notwendige Changes.

Eine Maßnahme mit hoher Risikoreduktion kann wichtiger sein als eine einfache technische Optimierung.

Wirksamkeit prüfen

Nach Umsetzung einer Maßnahme sollte geprüft werden:

- Tritt der Incident erneut auf?
- Sind ähnliche Incidents zurückgegangen?
- Funktioniert der Workaround?
- Hat Monitoring früher gewarnt?
- Sind Tickets vollständiger?
- Wurde die Eskalation schneller?
- Hat sich die Benutzererfahrung verbessert?
- Gab es neue Nebenwirkungen?

Ursachenanalyse ist erst dann wirklich abgeschlossen, wenn die Verbesserung überprüft wurde oder ein verbleibendes Risiko bewusst akzeptiert ist.

RCA-Ergebnis dokumentieren

Eine Abschlussdokumentation kann enthalten:

- Problemzusammenfassung,
- betroffene Services,
- Zeitraum,
- Auswirkungen,
- zugehörige Incidents,
- erkannte Ursachen,
- beitragende Faktoren,
- durchgeführte Tests,
- bestätigter Workaround,
- dauerhafte Lösung,
- offene Risiken,

- Maßnahmen,
- Verantwortliche,
- Lessons Learned,
- benötigte Changes,
- aktualisierte Knowledge-Artikel.

Die Dokumentation muss nicht übermäßig lang sein.

Sie muss nachvollziehbar und nützlich sein.

Praxisbeispiel: Zertifikat abgelaufen

Incident

Benutzer können sich nicht an einer Anwendung anmelden.

Direkte Ursache

Das Zertifikat des Anmeldedienstes ist abgelaufen.

Grundursache

Es gab keinen Prozess zur rechtzeitigen Zertifikatserneuerung.

Beitragende Faktoren

- kein Monitoring auf Zertifikatsablauf,
- keine verantwortliche Rolle,
- keine Erinnerungsfristen,
- keine Dokumentation im Betriebshandbuch.

Maßnahmen

- Zertifikatsmonitoring einführen,
- Owner festlegen,
- Ablaufdatum in Wartungskalender aufnehmen,
- Runbook zur Erneuerung erstellen,
- Knowledge-Artikel für Symptome aktualisieren.

Praxisbeispiel: Speicher läuft voll

Incident

Ein Dateiserver ist nicht mehr beschreibbar.

Workaround

Temporäre Dateien werden kontrolliert entfernt.

Ursachenanalyse

- Logdateien wachsen ungewöhnlich stark.
- Logrotation greift nicht.
- Monitoring warnt erst bei 98 Prozent.
- Verantwortlicher erhält keine Benachrichtigung.

Maßnahmen

- Logrotation korrigieren,
- Monitoringgrenzen anpassen,
- Alarmempfänger aktualisieren,
- Kapazitätsbericht ergänzen,
- Runbook für Speicherwarnungen erstellen.

Praxisbeispiel: Wiederkehrende VPN-Abbrüche

Problem

Mehrere Benutzer verlieren nach dem Ruhezustand die VPN-Verbindung.

Fakten

- nur Windows-Notebooks betroffen,
- nur Client-Version 5.8 betroffen,
- Fehler tritt nach Ruhezustand auf,
- Neustart des Clients hilft kurzfristig,
- Hersteller bestätigt bekannten Fehler.

Known Error

Client-Version 5.8 verursacht nach Ruhezustand fehlerhafte Tunnelzustände.

Workaround

VPN-Client vollständig beenden und neu starten.

Dauerhafte Lösung

Test und Ausrollen einer korrigierten Client-Version über Change Enablement.

Typische Fehler

Fehler 1

Symptom wird als Grundursache dokumentiert.

Fehler 2

Die erste Vermutung wird nicht mehr überprüft.

Fehler 3

Nur technische Ursachen werden betrachtet.

Fehler 4

Organisatorische und prozessuale Faktoren werden ignoriert.

Fehler 5

Zeitlicher Zusammenhang wird als Beweis behandelt.

Fehler 6

Mehrere Änderungen werden gleichzeitig getestet.

Fehler 7

RCA wird zur Schuldzuweisung verwendet.

Fehler 8

Workaround wird nicht dokumentiert.

Fehler 9

Maßnahmen werden beschlossen, aber nicht verfolgt.

Fehler 10

Lessons Learned werden nicht in Knowledge, Monitoring oder Prozesse übernommen.

Fehler 11

CMDB- und Serviceabhängigkeiten werden nicht genutzt.

Fehler 12

Wirksamkeit der Maßnahmen wird nicht geprüft.

Checkliste Ursachenanalyse starten

- [ ] Problem eindeutig beschrieben
- [ ] betroffene Services bekannt
- [ ] zugehörige Incidents verknüpft
- [ ] Auswirkungen dokumentiert
- [ ] Zeitlinie begonnen
- [ ] relevante Changes geprüft
- [ ] Monitoringdaten gesichert
- [ ] Logs gesichert
- [ ] Workaround bekannt oder gesucht
- [ ] Verantwortlicher für Analyse benannt

Checkliste Ursachen untersuchen

- [ ] Symptome von Ursachen getrennt
- [ ] Hypothesen dokumentiert
- [ ] Hypothesen überprüfbar formuliert
- [ ] Fakten und Vermutungen getrennt
- [ ] Umfang geprüft
- [ ] betroffene und nicht betroffene Bereiche verglichen
- [ ] technische Ursachen geprüft
- [ ] Prozessursachen geprüft
- [ ] organisatorische Faktoren geprüft
- [ ] menschliche Faktoren berücksichtigt
- [ ] beitragende Faktoren dokumentiert

Checkliste Maßnahmen ableiten

- [ ] Workaround dokumentiert

- [ ] Known Error bewertet
- [ ] dauerhafte Lösung beschrieben
- [ ] notwendiger Change geprüft
- [ ] Risiken bewertet
- [ ] Maßnahmen priorisiert
- [ ] Verantwortliche benannt
- [ ] Termine festgelegt
- [ ] Knowledge-Artikel aktualisiert
- [ ] Monitoring oder Runbook angepasst
- [ ] Wirksamkeitsprüfung geplant

Bedeutung für Fachinformatiker für Systemintegration

Für Fachinformatiker ist Ursachenanalyse eine zentrale Fähigkeit.

Im Arbeitsalltag bedeutet das:

- nicht bei der ersten Vermutung stehen bleiben,
- technische Beobachtungen sauber dokumentieren,
- Logs und Messwerte sinnvoll auswerten,
- Änderungen und Abhängigkeiten prüfen,
- wiederkehrende Muster erkennen,
- Workarounds nutzbar dokumentieren,
- Risiken einschätzen,
- und Verbesserungen anstoßen.

Gute Ursachenanalyse macht Systeme nicht nur wieder lauffähig.

Sie macht sie langfristig stabiler.

Zusammenfassung

> Symptom erkennen
> ↓
> Problem beschreiben
> ↓
> Fakten sammeln
> ↓
> Zeitlinie und Umfang prüfen
> ↓

> Hypothesen bilden
> ↓
> Hypothesen testen
> ↓
> technische, prozessuale und organisatorische Ursachen betrachten
> ↓
> Workaround und Known Error dokumentieren
> ↓
> dauerhafte Maßnahmen ableiten
> ↓
> Wirksamkeit prüfen
> ↓
> Wissen und Verbesserungen übernehmen

Merksätze

> Die erste Erklärung ist nicht immer die richtige Ursache.

> Ein Symptom ist noch keine Grundursache.

> Ursachenanalyse ist Lernen, nicht Schuldzuweisung.

> Viele IT-Probleme haben technische und organisatorische Ursachen.

> Workarounds helfen kurzfristig, ersetzen aber keine dauerhafte Lösung.

> Eine RCA ist nur wertvoll, wenn daraus konkrete Verbesserungen entstehen.

Verwandte Seiten

- 4.1 Problem Management – Ziele, Begriffe und Abgrenzung
- 4.3 Workarounds und Known Errors
- 4.4 Trendanalyse und proaktives Problem Management
- 4.5 Problem Management im Zusammenspiel mit Incident, Change und Knowledge Management
- Incident Management
- Change Enablement
- Knowledge Management
- Service Configuration Management
- Continual Improvement

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Problem Management
- PeopleCert – ITIL Practice Guide: Incident Management
- PeopleCert – ITIL Practice Guide: Knowledge Management
- PeopleCert – ITIL Practice Guide: Change Enablement
- PeopleCert – ITIL Practice Guide: Service Configuration Management
- ITIL Foundation – Version 5

Einordnung

Die dargestellten:

- RCA-Methoden,
- 5-Why-Beispiele,
- Ishikawa-Kategorien,
- Pareto-Beispiele,
- Checklisten,
- Ursachenarten,
- und Praxisbeispiele

sind herstellerneutrale Praxisempfehlungen.

ITIL schreibt keine universelle:

- RCA-Methode,
- Anzahl von Analysefragen,
- Diagrammform,
- Review-Struktur,
- Maßnahmenmatrix,
- oder konkrete Dokumentationsvorlage

für alle Organisationen vor.

Die konkrete Umsetzung muss an:

- Services,
- Risiken,
- technische Umgebung,
- Datenqualität,
- Organisation,
- Lieferanten,
- Service Levels,
- Sicherheitsanforderungen,
- und verfügbare Fähigkeiten

angepasst werden.

Behandelter Framework-Stand: ITIL Version 5
Zusätzlich berücksichtigt: aktuelle ITIL-4-Practice-Guidance
Fachlicher Stand: August 2026
