# 4.1 Problem Management – Ziele, Begriffe und Abgrenzung

> **Kurz erklärt**
>
> Problem Management beschäftigt sich mit den Ursachen von Incidents.
>
> Während Incident Management vor allem den Service schnell wiederherstellen soll, untersucht Problem Management, warum Incidents entstehen und wie Wiederholungen vermieden werden können.
>
> Ziel ist nicht nur die schnelle Reparatur, sondern eine nachhaltige Verbesserung der Servicequalität.

---

### Warum Problem Management wichtig ist

Viele Störungen treten nicht nur einmal auf.

Beispiele:

- VPN bricht regelmäßig ab.
- Druckerwarteschlangen blockieren immer wieder.
- Benutzer melden wiederholt Anmeldeprobleme.
- Ein Dienst muss regelmäßig neu gestartet werden.
- Nach Updates entstehen ähnliche Fehler.
- Ein Speicherbereich läuft immer wieder voll.

Wenn solche Fälle nur einzeln gelöst werden, bleibt die eigentliche Ursache bestehen.

Problem Management hilft dabei:

- wiederkehrende Incidents zu erkennen,
- Ursachen systematisch zu untersuchen,
- Workarounds bereitzustellen,
- dauerhafte Lösungen vorzubereiten,
- bekannte Fehler zu dokumentieren,
- und zukünftige Incidents zu vermeiden.

---

### Incident und Problem unterscheiden

| Incident | Problem |
| --- | --- |

| | |
|---|---|
| ungeplante Unterbrechung oder Qualitätsminderung eines Service | Ursache oder mögliche Ursache eines oder mehrerer Incidents |
| Fokus auf schnelle Wiederherstellung | Fokus auf Ursachenverständnis und Vermeidung |
| kurzfristige Bearbeitung | häufig längerfristige Analyse |
| Workaround kann ausreichen | dauerhafte Lösung wird angestrebt |
| Ziel: Service wieder nutzbar machen | Ziel: Wiederholung verhindern oder Auswirkungen reduzieren |

Beispiel:

**Incident**

Benutzer können sich nicht am VPN anmelden.

**Problem**

Die Ursache liegt in einer fehlerhaften Zertifikatsverteilung, die regelmäßig zu abgelaufenen Client-Zertifikaten führt.

---

### Problem Management ist nicht nur Fehlerbehebung

Problem Management bedeutet nicht nur:

> “ Jemand sucht den technischen Fehler.

Es umfasst auch:

- Muster erkennen,
- Daten auswerten,
- Risiken bewerten,
- bekannte Fehler dokumentieren,
- Workarounds bereitstellen,
- Änderungen vorbereiten,
- Wissen nutzbar machen,
- und Verbesserungen nachverfolgen.

Ein Problem kann auch dann angelegt werden, wenn noch keine endgültige Ursache bekannt ist.

---

### Problem, Known Error und Workaround unterscheiden

| Begriff | Bedeutung |
|---|---|

| | |
|---|---|
| **Problem** | Ursache oder mögliche Ursache eines oder mehrerer Incidents |
| **Known Error** | Problem mit bekannter Ursache oder bekanntem Fehlerzustand |
| **Workaround** | vorübergehende Möglichkeit, Auswirkungen zu umgehen oder zu reduzieren |
| **dauerhafte Lösung** | Maßnahme, die die Ursache beseitigt oder das Risiko nachhaltig reduziert |

Beispiel:

**Problem**

Mehrere Clients verlieren nach dem Ruhezustand die VPN-Verbindung.

**Known Error**

Bestimmte Client-Version verursacht nach dem Ruhezustand fehlerhafte Tunnelzustände.

**Workaround**

VPN-Client vollständig beenden und neu starten.

**Dauerhafte Lösung**

Client-Version aktualisieren oder Konfiguration ändern.

---

**Problem Management und Incident Management arbeiten zusammen**

Incident Management liefert wichtige Informationen für Problem Management.

Dazu gehören:

- betroffene Services,
- Symptome,
- Zeitpunkte,
- Häufigkeit,
- betroffene Benutzer,
- Workarounds,
- technische Maßnahmen,
- Logs,
- und wiederkehrende Muster.

Problem Management liefert zurück:

- Known Errors,

- Workarounds,
- Ursacheninformationen,
- Empfehlungen,
- Knowledge-Artikel,
- und Change-Vorschläge.

Dadurch werden zukünftige Incidents schneller lösbar oder vollständig vermieden.

---

**Reaktives Problem Management**

Reaktives Problem Management beginnt nach Incidents.

Auslöser können sein:

- wiederkehrende Incidents,
- Major Incidents,
- hohe Auswirkungen,
- unbekannte Ursache,
- viele gleichartige Meldungen,
- häufige Workarounds,
- auffällige Eskalationen,
- oder wiederholte SLA-Gefährdung.

Beispiel:

Nach einem Major Incident wird untersucht, warum ein zentraler Dienst ausgefallen ist und warum Monitoring nicht früher gewarnt hat.

---

**Proaktives Problem Management**

Proaktives Problem Management sucht nach Problemen, bevor größere Incidents entstehen.

Informationsquellen können sein:

- Trends,
- Monitoringdaten,
- Kapazitätsdaten,
- Loganalysen,
- Service Reviews,
- Sicherheitsmeldungen,
- Benutzerfeedback,
- wiederkehrende kleine Störungen,
- technische Schulden,
- und Lieferanteninformationen.

Beispiel:

Monitoring zeigt, dass Speicherplatz auf mehreren Servern regelmäßig kritisch wird.

Noch ist kein Ausfall eingetreten.

Problem Management kann trotzdem prüfen, warum Kapazitätsplanung, Alarmierung oder Bereinigung nicht ausreichen.

---

**Typische Auslöser für einen Problem Record**

Ein Problem Record kann sinnvoll sein, wenn:

- mehrere ähnliche Incidents auftreten,
- ein Major Incident stattgefunden hat,
- die Ursache eines wichtigen Incidents unbekannt bleibt,
- ein Workaround häufig genutzt werden muss,
- ein Service wiederholt instabil ist,
- ein technisches Risiko erkennbar wird,
- ein Lieferant einen bekannten Fehler bestätigt,
- ein Sicherheitsrisiko vermutet wird,
- oder ein Trend auf zukünftige Störungen hindeutet.

Nicht jeder einzelne Incident benötigt automatisch einen Problem Record.

---

**Problem Record**

Ein Problem Record dokumentiert die Bearbeitung eines Problems.

Typische Inhalte:

- Beschreibung des Problems,
- betroffener Service,
- betroffene Configuration Items,
- zugehörige Incidents,
- Auswirkungen,
- Priorität,
- bekannte Symptome,
- Hypothesen,
- Untersuchungsergebnisse,
- Ursache, falls bekannt,
- Workaround,
- Known-Error-Status,
- empfohlene dauerhafte Lösung,
- erforderlicher Change,

- Verantwortlicher,
- Status,
- Lessons Learned.

Das konkrete Format hängt vom verwendeten ITSM-System und der Organisation ab.

---

**Priorisierung von Problems**

Problems sollten priorisiert werden.

Kriterien können sein:

- Anzahl betroffener Incidents,
- Auswirkung auf Services,
- geschäftliche Kritikalität,
- Sicherheitsrisiko,
- Häufigkeit,
- Kosten,
- Aufwand,
- Risiko einer Wiederholung,
- Verfügbarkeit eines Workarounds,
- und Nutzen einer dauerhaften Lösung.

Ein Problem mit wenigen Incidents kann trotzdem hohe Priorität besitzen, wenn es einen kritischen Service oder ein hohes Sicherheitsrisiko betrifft.

---

**Problem Management und Change Enablement**

Dauerhafte Lösungen erfordern häufig Änderungen.

Beispiele:

- Softwareupdate,
- Konfigurationsänderung,
- Austausch defekter Hardware,
- Änderung einer Firewall-Regel,
- Anpassung eines Prozesses,
- neues Monitoring,
- Änderung eines Deployments,
- Architekturverbesserung.

Solche Maßnahmen sollten nicht unkontrolliert umgesetzt werden.

Sie müssen entsprechend Risiko, Auswirkung und Organisationsregeln über Change Enablement gesteuert werden.

---

**Problem Management und Knowledge Management**

Problem Management erzeugt wertvolles Wissen.

Dieses Wissen sollte nutzbar gemacht werden für:

- Service Desk,
- Fachgruppen,
- Benutzer,
- Major-Incident-Bearbeitung,
- Self-Service,
- Runbooks,
- Known-Error-Datenbank,
- und Continual Improvement.

Beispiele:

- bekannter Fehler mit Workaround,
- Diagnosehinweise,
- Eskalationskriterien,
- betroffene Versionen,
- Abbruchkriterien,
- dauerhafte Lösung nach Change.

Wissen darf nicht nur im Kopf einzelner Spezialisten bleiben.

---

**Problem Management und Service Configuration Management**

Für Ursachenanalysen sind Service- und CI-Informationen wichtig.

Hilfreich sind:

- betroffene Configuration Items,
- Abhängigkeiten zwischen Services,
- installierte Versionen,
- letzte Änderungen,
- Standorte,
- Verantwortliche,
- Lieferanten,
- und technische Beziehungen.

Ohne diese Informationen wird Ursachenanalyse oft langsam und ungenau.

Beispiel:

Mehrere Anwendungen fallen aus.

Erst durch die CI-Beziehungen wird erkennbar, dass alle vom gleichen Datenbankcluster abhängig sind.

---

**Problem Management und Continual Improvement**

Problem Management ist eng mit kontinuierlicher Verbesserung verbunden.

Aus Problems können entstehen:

- Verbesserungsmaßnahmen,
- neue Standards,
- bessere Überwachung,
- aktualisierte Runbooks,
- Knowledge-Artikel,
- Schulungen,
- Automatisierungen,
- Architekturänderungen,
- oder Prozessanpassungen.

Ein gelöstes Problem sollte deshalb nicht nur abgeschlossen werden.

Es sollte geprüft werden, welche Erkenntnisse für zukünftige Arbeit nutzbar sind.

---

**Problem Management und Risiko**

Nicht jede Ursache kann sofort beseitigt werden.

Manchmal ist eine dauerhafte Lösung:

- zu teuer,
- zu riskant,
- technisch noch nicht verfügbar,
- abhängig von einem Lieferanten,
- nur in einem Wartungsfenster möglich,
- oder organisatorisch noch nicht entschieden.

Dann muss das verbleibende Risiko bewusst bewertet und dokumentiert werden.

Mögliche Maßnahmen:

- Workaround bereitstellen,
- Monitoring verbessern,
- Benutzer informieren,
- Risiko akzeptieren,
- Change später planen,

- Lieferant einbinden,
- oder Service-Level-Erwartungen anpassen.

---

**Workaround als wichtiger Zwischenschritt**

Ein Workaround ist keine endgültige Lösung, kann aber sehr wertvoll sein.

Er hilft dabei:

- Incidents schneller zu lösen,
- Auswirkungen zu reduzieren,
- Benutzer arbeitsfähig zu halten,
- Zeit für Ursachenanalyse zu gewinnen,
- Service Desk zu unterstützen.

Ein Workaround sollte dokumentiert werden mit:

- Anwendungsbereich,
- Voraussetzungen,
- Arbeitsschritten,
- Risiken,
- Einschränkungen,
- Gültigkeitsdauer,
- und Eskalationshinweisen.

---

**Dauerhafte Lösung**

Eine dauerhafte Lösung soll die Ursache beseitigen oder das Risiko wesentlich reduzieren.

Beispiele:

- fehlerhafte Softwareversion aktualisieren,
- Zertifikatsüberwachung einführen,
- Kapazität erweitern,
- defekte Hardware austauschen,
- Berechtigungskonzept korrigieren,
- Prozesslücke schließen,
- Monitoring verbessern,
- automatisierte Prüfung einführen,
- Schulung durchführen.

Nicht jede dauerhafte Lösung ist rein technisch.

Auch Prozesse, Kommunikation und Verantwortlichkeiten können Ursachen sein.

---

### Technische Ursache und organisatorische Ursache

Ein Problem kann mehrere Ursachenebenen besitzen.

Beispiel:

Ein Zertifikat läuft ab.

**Technische Ursache**

Zertifikat ist nicht mehr gültig.

**Organisatorische Ursache**

Es gab keinen verantwortlichen Owner für Zertifikatsüberwachung.

**Prozessursache**

Es gab keinen geregelten Ablauf für rechtzeitige Erneuerung.

**Verbesserung**

Monitoring, Verantwortlichkeit und Erneuerungsprozess werden eingeführt.

> **Merke**
>
> Die technische Ursache ist oft nur ein Teil der Gesamtursache.

---

### Problem Management ist keine Schuldsuche

Problem Management soll nicht klären, wer schuld ist.

Ziel ist:

- verstehen,
- lernen,
- verbessern,
- Wiederholungen vermeiden,
- Risiken reduzieren,
- Servicequalität erhöhen.

Eine Schuldzuweisung führt häufig dazu, dass Informationen zurückgehalten werden.

Eine lernorientierte Analyse führt zu besseren Ergebnissen.

---

**Praxisbeispiel: Wiederkehrende VPN-Störungen**

**Ausgangslage**

Mehrere Benutzer melden regelmäßig VPN-Abbrüche.

Incident Management stellt den Zugang jeweils durch Neustart des Clients wieder her.

**Problem Management**

Die Incidents werden gemeinsam ausgewertet.

Auffällig ist:

- gleiche Client-Version,
- gleiche Fehlermeldung,
- besonders häufig nach Ruhezustand,
- Workaround immer identisch.

**Ergebnis**

Ein Known Error wird dokumentiert.

Ein Workaround wird für den Service Desk bereitgestellt.

Ein Change zur Aktualisierung des VPN-Clients wird geplant.

---

**Praxisbeispiel: Druckerwarteschlange blockiert**

**Ausgangslage**

Ein Etikettendrucker im Versand blockiert mehrfach pro Woche.

Incident Management entfernt jeweils den fehlerhaften Druckauftrag.

**Problem Management**

Die Analyse zeigt:

- bestimmte PDF-Dateien verursachen den Fehler,
- Treiber ist veraltet,
- Workaround ist bekannt,
- Versand ist regelmäßig betroffen.

**Dauerhafte Lösung**

Treiber wird getestet und über Change Enablement aktualisiert.

Zusätzlich wird ein Knowledge-Artikel für den Service Desk erstellt.

---

**Praxisbeispiel: Speicher läuft voll**

**Ausgangslage**

Ein Server erzeugt regelmäßig Incidents wegen vollem Speicher.

Incident Management löscht temporäre Dateien.

**Problem Management**

Die Untersuchung zeigt:

- Logrotation funktioniert nicht korrekt,
- Monitoring warnt zu spät,
- Verantwortlichkeit für Logwachstum war unklar.

**Verbesserung**

- Logrotation korrigieren,
- Monitoring-Grenzwerte anpassen,
- Verantwortlichen festlegen,
- Runbook aktualisieren.

---

**Typische Fehler**

**Fehler 1**

Jeder Incident wird einzeln bearbeitet, ohne Muster zu erkennen.

---

**Fehler 2**

Problem Management wird erst nach sehr großen Störungen genutzt.

---

**Fehler 3**

Workarounds werden nicht dokumentiert.

---

**Fehler 4**

Known Errors bleiben nur einzelnen Spezialisten bekannt.

---

**Fehler 5**

Die technische Ursache wird gefunden, aber organisatorische Ursachen werden ignoriert.

---

**Fehler 6**

Dauerhafte Lösungen werden ohne Change-Bewertung umgesetzt.

---

**Fehler 7**

Problems werden eröffnet, aber nicht aktiv verfolgt.

---

**Fehler 8**

Priorisierung fehlt.

---

**Fehler 9**

Problem Management wird als Schuldsuche verstanden.

---

**Fehler 10**

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

---

**Checkliste Problem erfassen**

- [ ] Wiederkehrende oder schwerwiegende Incidents geprüft
- [ ] betroffener Service beschrieben
- [ ] zugehörige Incidents verknüpft
- [ ] Auswirkungen dokumentiert
- [ ] Symptome zusammengefasst
- [ ] erste Hypothesen dokumentiert
- [ ] Priorität bewertet
- [ ] Verantwortlicher benannt
- [ ] Workaround geprüft
- [ ] Knowledge-Bedarf erkannt

---

**Checkliste Problem bearbeiten**

- [ ] relevante Daten gesammelt
- [ ] Incidents verglichen
- [ ] Changes geprüft
- [ ] Logs und Monitoringdaten ausgewertet
- [ ] betroffene CIs geprüft
- [ ] Ursachenhypothesen getestet
- [ ] Workaround dokumentiert
- [ ] Known Error bewertet
- [ ] dauerhafte Lösung vorgeschlagen
- [ ] Change-Bedarf geprüft
- [ ] Risiko dokumentiert
- [ ] Stakeholder informiert

---

**Checkliste Problem abschließen**

- [ ] Ursache ausreichend verstanden oder Risiko bewertet
- [ ] Workaround dokumentiert
- [ ] Known Error aktualisiert
- [ ] dauerhafte Lösung umgesetzt oder bewusst geplant
- [ ] zugehörige Incidents berücksichtigt
- [ ] Knowledge-Artikel erstellt oder aktualisiert
- [ ] Monitoring oder Runbooks angepasst
- [ ] Lessons Learned dokumentiert
- [ ] Verbesserungsmaßnahmen verfolgt
- [ ] Rest Risiko bekannt und akzeptiert

---

**Bedeutung für Fachinformatiker für Systemintegration**

Fachinformatiker für Systemintegration liefern häufig die wichtigsten technischen Informationen für Problem Management.

Dazu gehören:

- Logs,
- Messwerte,
- Konfigurationen,
- Versionsstände,
- Abhängigkeiten,

- Change-Historie,
- Workarounds,
- Fehlerhäufigkeit,
- und technische Hypothesen.

Wichtig ist nicht nur, einen Incident schnell zu beheben.

Wichtig ist auch, wiederkehrende Ursachen zu erkennen und die Umgebung dauerhaft stabiler zu machen.

---

**Zusammenfassung**

> Incidents treten auf
> ↓
> Muster oder schwere Auswirkungen werden erkannt
> ↓
> Problem Record wird erstellt
> ↓
> Ursache oder mögliche Ursache wird untersucht
> ↓
> Workaround wird dokumentiert
> ↓
> Known Error wird bei Bedarf erfasst
> ↓
> dauerhafte Lösung wird geplant
> ↓
> Change, Knowledge und Improvement werden eingebunden
> ↓
> zukünftige Incidents werden reduziert oder schneller lösbar

---

**Merksätze**

> Incident Management stellt den Service wieder her.

> Problem Management versteht und reduziert Ursachen.

>

> Ein Workaround ist hilfreich, aber keine dauerhafte Lösung.

> Ein Known Error muss nutzbar dokumentiert sein.

> Problem Management ist Lernen, nicht Schuldsuche.

> Gute Problem Records machen Services langfristig stabiler.

---

**Verwandte Seiten**

- 3.6 Erstdiagnose, Lösung und funktionale Eskalation
- 3.7 Ownership, hierarchische Eskalation und Major Incidents
- 3.9 Kennzahlen, Qualität und Continual Improvement im Service Desk
- 4.2 Ursachenanalyse
- 4.3 Workarounds und Known Errors
- 4.4 Trendanalyse und proaktives Problem 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:

- Problem-Record-Inhalte,

- Beispiele,
- Checklisten,
- Priorisierungskriterien,
- Workaround-Hinweise,
- und Praxisabläufe

sind herstellerneutrale Praxisempfehlungen.

ITIL schreibt keine universelle:

- Problem-Record-Vorlage,
- RCA-Methode,
- Prioritätsmatrix,
- Workaround-Struktur,
- Review-Vorgabe,
- oder konkrete Toolauswahl

für alle Organisationen vor.

Die konkrete Umsetzung muss an:

- Services,
- Risiken,
- Organisation,
- Datenqualität,
- Supportmodell,
- Lieferanten,
- Service Levels,
- und verfügbare Fähigkeiten

angepasst werden.

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

---