4.5 Problem Management im Zusammenspiel mit Incident, Change und Knowledge Management

> **Kurz erklärt**
>
> Problem Management arbeitet nicht isoliert.
>
> Es ist eng verbunden mit Incident Management, Change Enablement, Knowledge Management, Service Configuration Management und Continual Improvement.
>
> Nur wenn diese Practices zusammenarbeiten, können Incidents schnell bearbeitet, Ursachen verstanden, dauerhafte Lösungen umgesetzt und Wissen für zukünftige Fälle nutzbar gemacht werden.

Warum das Zusammenspiel wichtig ist

Ein Problem entsteht häufig aus wiederkehrenden oder schwerwiegenden Incidents.

Die dauerhafte Lösung benötigt oft einen Change.

Der Service Desk benötigt Workarounds und Known Errors aus dem Knowledge Management.

Service Configuration Management liefert Informationen über betroffene Systeme und Abhängigkeiten.

Continual Improvement sorgt dafür, dass Erkenntnisse nicht verloren gehen.

Ohne dieses Zusammenspiel entstehen typische Schwächen:

- Incidents werden einzeln gelöst, aber Ursachen bleiben bestehen.
- Workarounds sind bekannt, aber nicht dokumentiert.
- Changes werden durchgeführt, ohne das Problem vollständig zu verstehen.
- Knowledge-Artikel werden nicht aktualisiert.
- Serviceabhängigkeiten sind unbekannt.
- Lessons Learned werden nicht umgesetzt.
- dieselben Incidents treten immer wieder auf.

Überblick über das Zusammenspiel

```
Incident tritt auf
        ↓
Incident Management stellt Service wieder her
        ↓
wiederkehrendes Muster oder schwere Auswirkung wird erkannt
        ↓
Problem Management untersucht Ursache
        ↓
Workaround oder Known Error wird dokumentiert
        ↓
Knowledge Management stellt Wissen bereit
        ↓
dauerhafte Lösung wird vorbereitet
        ↓
Change Enablement steuert Umsetzung
        ↓
Service wird stabilisiert
        ↓
Continual Improvement übernimmt Erkenntnisse
```

Problem Management und Incident Management

Incident Management und Problem Management haben unterschiedliche Schwerpunkte.

Incident Management	Problem Management
schnelle Wiederherstellung des Service	Ursache verstehen und Wiederholung vermeiden
kurzfristiger Fokus	nachhaltiger Fokus
einzelne Störung bearbeiten	Muster und Ursachen untersuchen
Workaround kann genügen	dauerhafte Lösung wird angestrebt
Benutzer schnell arbeitsfähig machen	Service langfristig stabiler machen

Beide Practices ergänzen sich.

Incident Management liefert Daten und Erfahrungen.

Problem Management liefert Workarounds, Known Errors und dauerhafte Verbesserungen zurück.

Was Incident Management an Problem Management liefert

Wichtige Informationen aus Incidents:

- betroffener Service,
- Symptome,
- Fehlermeldungen,
- Zeitpunkt,
- betroffene Benutzer oder Standorte,
- Priorität,
- Auswirkungen,
- erste Diagnose,
- durchgeführte Maßnahmen,
- verwendete Workarounds,
- Eskalationen,
- Logs,
- Monitoringdaten,
- betroffene Configuration Items,
- Benutzerfeedback,
- Wiedereröffnungen.

Je besser Incidents dokumentiert sind, desto leichter kann Problem Management Muster erkennen.

Was Problem Management an Incident Management zurückliefert

Problem Management unterstützt Incident Management durch:

- Known Errors,
- Workarounds,
- Diagnosehinweise,
- Eskalationsregeln,
- Knowledge-Artikel,
- Runbooks,
- Hinweise auf betroffene Versionen,
- bekannte Symptome,
- bekannte Risiken,
- und geplante dauerhafte Lösungen.

Dadurch kann der Service Desk zukünftige Incidents schneller erkennen und bearbeiten.

Beispiel:

Wenn ein VPN-Fehler als Known Error dokumentiert ist, muss der Service Desk nicht bei jedem neuen Ticket von vorn analysieren.

Wann ein Incident zu einem Problem führen kann

Ein Problem Record kann sinnvoll sein, wenn:

- mehrere ähnliche Incidents auftreten,
- ein Major Incident stattgefunden hat,
- die Ursache unbekannt bleibt,
- ein Workaround häufig genutzt wird,
- ein Service wiederholt instabil ist,
- ein Sicherheitsrisiko vermutet wird,
- ein Trend erkennbar ist,
- oder ein einzelner Incident sehr hohe Auswirkungen hatte.

Nicht jeder Incident benötigt automatisch ein Problem.

Die Organisation sollte Kriterien festlegen.

Beispiel: Incident zu Problem

Incident

Mehrere Benutzer melden, dass VPN nach dem Ruhezustand nicht mehr funktioniert.

Incident Management

Der Service Desk stellt die Verbindung durch Neustart des VPN-Clients wieder her.

Problem Management

Mehrere gleichartige Incidents werden verglichen.

Es wird erkannt:

- gleiche Client-Version,
- gleiches Fehlerbild,
- gleicher Auslöser,
- gleicher Workaround.

Ergebnis

Ein Problem Record wird erstellt.

Ein Known Error und ein Workaround werden dokumentiert.

Ein Change für eine neue Client-Version wird vorbereitet.

Problem Management und Change Enablement

Dauerhafte Lösungen erfordern häufig Änderungen an produktiven Systemen.

Beispiele:

- Softwareupdate,
- Konfigurationsänderung,
- Austausch defekter Hardware,
- Anpassung einer Firewall-Regel,
- Änderung eines Deployments,
- Zertifikatserneuerung,
- Monitoring-Erweiterung,
- Automatisierung,
- Prozessänderung.

Solche Maßnahmen dürfen nicht unkontrolliert durchgeführt werden.

Change Enablement sorgt dafür, dass Änderungen bewertet, geplant, genehmigt, umgesetzt und überprüft werden.

Warum dauerhafte Lösungen oft Changes sind

Eine dauerhafte Lösung verändert häufig den Zustand einer Umgebung.

Beispiele:

Problem	mögliche dauerhafte Lösung	Bezug zu Change Enablement
VPN-Client fehlerhaft	neue Version ausrollen	Software-Change
Zertifikat läuft ab	Zertifikat erneuern und Monitoring ergänzen	technischer Change
Speicher läuft voll	Logrotation ändern	Konfigurations-Change
Druckertreiber fehlerhaft	Treiber aktualisieren	Standard- oder Normal-Change
Dienst stürzt regelmäßig ab	Anwendungspatch einspielen	Release oder Change
Berechtigungen falsch modelliert	Rollenkonzept anpassen	organisatorischer und technischer Change

Die Art des Changes hängt von Risiko, Auswirkung und Organisationsregeln ab.

Problem Management liefert Input für Changes

Ein guter Change zur Problemlösung sollte auf nachvollziehbaren Informationen beruhen.

Problem Management liefert dafür:

- Problembeschreibung,
- Ursache oder wahrscheinliche Ursache,
- Auswirkungen,
- betroffene Services,
- betroffene Configuration Items,
- Risiko bei Nicht-Handeln,
- Workaround,
- bekannte Einschränkungen,
- empfohlene Lösung,
- erwarteten Nutzen,
- mögliche Nebenwirkungen,
- Testhinweise,
- Rollback-Hinweise,
- Verknüpfung zu Incidents.

Dadurch kann Change Enablement besser bewerten, ob und wie die Änderung umgesetzt werden soll.

Change Enablement liefert Informationen zurück

Nach einem Change sollte geprüft werden:

- Wurde die dauerhafte Lösung erfolgreich umgesetzt?
- Sind die betroffenen Incidents zurückgegangen?
- Sind neue Fehler entstanden?
- Funktioniert der Service stabil?
- Muss der Workaround angepasst oder entfernt werden?
- Muss der Known Error geschlossen werden?
- Müssen Knowledge-Artikel aktualisiert werden?
- Muss Monitoring angepasst werden?

Ein Problem sollte nicht automatisch geschlossen werden, nur weil ein Change umgesetzt wurde.

Die Wirksamkeit muss überprüft werden.

Emergency Change und Problem Management

Bei sehr kritischen Situationen kann eine schnelle Änderung notwendig sein.

Beispiele:

- Sicherheitslücke wird aktiv ausgenutzt,
- zentraler Service ist ausgefallen,
- Datenverlust droht,
- ein Workaround ist nicht verfügbar,

- ein Major Incident erfordert sofortige technische Änderung.

Auch dann sollten Entscheidungen dokumentiert werden.

Nach der Wiederherstellung sollten Problem Management und Review klären:

- Warum war ein Emergency Change nötig?
- War die Änderung wirksam?
- Welche Risiken entstanden?
- Welche dauerhafte Lösung ist noch erforderlich?
- Wie kann eine ähnliche Situation künftig vermieden werden?

Problem Management und Knowledge Management

Problem Management erzeugt Wissen.

Dieses Wissen muss auffindbar, verständlich und aktuell sein.

Geeignete Inhalte für Knowledge Management:

- bekannte Symptome,
- Workarounds,
- Known Errors,
- Prüfschritte,
- Eskalationskriterien,
- betroffene Versionen,
- Abbruchkriterien,
- sichere Benutzeranleitungen,
- interne Runbooks,
- dauerhafte Lösungen,
- Lessons Learned.

Wenn Wissen nicht dokumentiert wird, müssen Teams dieselben Fälle immer wieder neu analysieren.

Unterschiedliche Zielgruppen für Wissen

Nicht jedes Wissen gehört in denselben Artikel.

Zielgruppe	Inhalt
Benutzer	einfache Anleitung, Workaround, Statushinweise
Service Desk	Prüfschritte, bekannte Symptome, Eskalationsregeln
Fachteam	technische Details, Logs, Konfigurationen, Ursachen

Zielgruppe	Inhalt
Management	Risiko, Auswirkungen, Verbesserungsstatus
Lieferant	Produktversion, Fehlerbild, technische Nachweise

Ein Benutzerartikel sollte keine internen Diagnoseschritte oder vertraulichen technischen Details enthalten.

Known Error als Wissensbaustein

Ein Known Error ist besonders wichtig für Knowledge Management.

Er beschreibt:

- welches Fehlerbild bekannt ist,
- welcher Service betroffen ist,
- welche Ursache oder wahrscheinliche Ursache vorliegt,
- welcher Workaround gilt,
- wann eskaliert werden muss,
- welche dauerhafte Lösung geplant ist,
- und ob ein Change offen ist.

Dadurch kann der Service Desk schneller entscheiden, ob ein neuer Incident zu einem bekannten Fehler passt.

Knowledge-Artikel aktuell halten

Nach Problem- und Change-Bearbeitung müssen Wissensartikel geprüft werden.

Zu aktualisieren sind möglicherweise:

- Workarounds,
- Known-Error-Status,
- Screenshots,
- Menüpfade,
- betroffene Versionen,
- Eskalationswege,
- Sicherheitswarnungen,
- Benutzerinformationen,
- Runbooks,
- Monitoringhinweise,
- Abschlusskriterien.

Ein gelöster Known Error mit veraltetem Workaround kann später neue Verwirrung erzeugen.

Problem Management und Service Configuration Management

Für Problem Management sind Informationen über Configuration Items und Serviceabhängigkeiten sehr wichtig.

Hilfreich sind:

- betroffene Server,
- Anwendungen,
- Datenbanken,
- Netzwerke,
- Zertifikate,
- Schnittstellen,
- Cloud-Dienste,
- Lieferantenservices,
- Versionen,
- Standorte,
- Verantwortliche,
- Abhängigkeiten.

Ohne diese Informationen ist schwer erkennbar, welche Komponenten gemeinsam betroffen sind.

Beispiel: Abhängigkeiten erkennen

Mehrere Anwendungen melden Verbindungsfehler.

Einzelne Teams prüfen zunächst ihre Anwendungen getrennt.

Durch Service- und CI-Beziehungen wird sichtbar:

- alle Anwendungen nutzen denselben Datenbankcluster,
- der Datenbankcluster nutzt denselben Storage,
- genau dieser Storage zeigt erhöhte Fehlerraten.

Problem Management kann dadurch gezielter analysieren.

Problem Management und Service Level Management

Service Level Management hilft zu bewerten, welche Problems besonders wichtig sind.

Zu berücksichtigen sind:

- vereinbarte Service Levels,
- betroffene Servicezeiten,
- geschäftskritische Services,
- wiederholte SLA-Gefährdung,

- Benutzer- oder Kundenimpact,
- Service Reviews,
- Beschwerden,
- vertragliche Verpflichtungen.

Ein Problem mit wenigen Incidents kann hohe Priorität besitzen, wenn ein kritischer Service oder ein wichtiges Service Level gefährdet ist.

Problem Management und Supplier Management

Viele Problems betreffen Lieferanten oder Hersteller.

Beispiele:

- Software-Bug,
- Cloud-Service-Störung,
- Firmwarefehler,
- Providerproblem,
- fehlerhafte Schnittstelle,
- Lizenzdienst nicht erreichbar,
- Hardwarefehler.

Problem Management sollte dokumentieren:

- Lieferantenticket,
- Herstellerreferenz,
- betroffene Version,
- empfohlener Workaround,
- geplante Korrektur,
- interne Auswirkungen,
- Eskalationsweg,
- offene Risiken.

Auch wenn ein Lieferant technisch verantwortlich ist, bleibt die interne Serviceverantwortung bestehen.

Problem Management und Continual Improvement

Problem Management liefert viele Verbesserungsideen.

Beispiele:

- Monitoring erweitern,
- Knowledge Base verbessern,
- Runbooks erstellen,

- Standard-Changes definieren,
- Schulungen durchführen,
- Automatisierung einführen,
- Serviceabhängigkeiten pflegen,
- Prozesse verbessern,
- Lieferantensteuerung anpassen.

Diese Verbesserungen sollten nicht nur mündlich besprochen werden.

Sie sollten in einem Improvement Register oder vergleichbaren System verfolgt werden.

Problem Management und Information Security Management

Manche Problems haben Sicherheitsbezug.

Beispiele:

- wiederkehrende kompromittierte Konten,
- unsichere Standardkonfiguration,
- veraltete Software,
- fehlende Patchprozesse,
- zu weitreichende Berechtigungen,
- unzureichende Protokollierung,
- wiederholte Phishing-Erfolge,
- fehlende MFA-Ausnahmenkontrolle.

Sicherheitsbezogene Problems benötigen möglicherweise:

- besondere Vertraulichkeit,
- andere Eskalationswege,
- Beweissicherung,
- Risikobewertung,
- Meldefristen,
- und Abstimmung mit Informationssicherheit.

Problem Management und Monitoring

Monitoring unterstützt Problem Management durch:

- Frühwarnungen,
- Trends,
- Fehlerraten,
- Kapazitätsdaten,
- Verfügbarkeitsdaten,
- Antwortzeiten,

- Logereignisse,
- Zertifikatsabläufe,
- Backupstatus,
- Hardwarezustände.

Problem Management kann umgekehrt Monitoring verbessern.

Beispiele:

- neuer Alarm für Zertifikatsablauf,
- zusätzliche Überwachung von Logwachstum,
- Dashboard für häufig betroffene Services,
- bessere Alarmgrenzen,
- Überwachung eines Workarounds,
- Prüfung nach Change.

Zusammenspiel als Kreislauf

```
Incident-Daten
    ↓
Problem-Analyse
    ↓
Known Error und Workaround
    ↓
Knowledge-Artikel
    ↓
dauerhafte Lösung
    ↓
Change
    ↓
Wirksamkeitsprüfung
    ↓
Verbesserung von Monitoring, Wissen und Prozessen
    ↓
weniger oder schneller lösbare Incidents
```

Typische Fehler im Zusammenspiel

Fehler 1

Incidents werden gelöst, aber nicht mit Problems verknüpft.

Fehler 2

Problem Management erhält zu wenig Informationen aus Tickets.

Fehler 3

Workarounds werden gefunden, aber nicht in Knowledge Management übernommen.

Fehler 4

Known Errors sind nur Spezialisten bekannt.

Fehler 5

Dauerhafte Lösungen werden ohne Change-Bewertung umgesetzt.

Fehler 6

Nach einem Change wird nicht geprüft, ob das Problem wirklich gelöst ist.

Fehler 7

Knowledge-Artikel bleiben nach einer dauerhaften Lösung unverändert.

Fehler 8

CMDB- und Serviceabhängigkeiten werden nicht genutzt.

Fehler 9

Lieferantenfehler werden intern nicht nachvollziehbar dokumentiert.

Fehler 10

Lessons Learned werden nicht in Continual Improvement übernommen.

Fehler 11

Sicherheitsbezug wird zu spät erkannt.

Fehler 12

Problem Records bleiben offen, ohne Verantwortlichen oder nächsten Schritt.

Praxisbeispiel: VPN-Problem

Incident Management

Mehrere Benutzer melden VPN-Abbrüche.

Der Service Desk stellt die Verbindung durch Neustart des Clients wieder her.

Problem Management

Incidents werden verglichen.

Die Analyse zeigt:

- gleiche Client-Version,
- gleiches Fehlerbild,
- Auftreten nach Ruhezustand,
- bekannter Herstellerfehler.

Knowledge Management

Ein Known Error und ein Workaround werden dokumentiert.

Der Service Desk erhält Prüfschritte.

Change Enablement

Eine neue Client-Version wird getestet und kontrolliert ausgerollt.

Continual Improvement

Der Updateprozess wird angepasst, damit Clientupdates künftig besser getestet werden.

Praxisbeispiel: Zertifikatsausfall

Incident Management

Benutzer können sich nicht an einer Anwendung anmelden.

Das abgelaufene Zertifikat wird erneuert.

Problem Management

Die Ursache wird untersucht.

Ergebnis:

- kein Monitoring für Zertifikatsablauf,
- kein Owner für Zertifikate,
- kein Runbook für Erneuerung.

Knowledge Management

Symptome und Prüfschritte werden dokumentiert.

Change Enablement

Monitoring und Erneuerungsprozess werden eingeführt.

Continual Improvement

Zertifikatsübersicht und Wartungskalender werden als Standard etabliert.

Praxisbeispiel: Druckerfehler

Incident Management

Etikettendrucker blockiert regelmäßig.

Der Service Desk leert die Warteschlange.

Problem Management

Bestimmte PDF-Dateien und ein alter Treiber werden als Ursache erkannt.

Knowledge Management

Workaround wird dokumentiert.

Change Enablement

Treiberupdate wird getestet und ausgerollt.

Service Desk

Künftige Incidents können schneller erkannt werden.

Continual Improvement

Druckertreiber werden in einen regelmäßigen Prüfprozess aufgenommen.

Checkliste Zusammenspiel mit Incident Management

- [ ] zugehörige Incidents verknüpft
- [ ] Symptome aus Incidents zusammengefasst
- [ ] Auswirkungen dokumentiert
- [ ] Workarounds aus Incidents geprüft
- [ ] Wiedereröffnungen berücksichtigt
- [ ] Eskalationen ausgewertet
- [ ] Service Desk über Known Error informiert
- [ ] Incident-Abschlusskriterien geprüft
- [ ] neue Incidents dem Problem zugeordnet
- [ ] Trenddaten berücksichtigt

Checkliste Zusammenspiel mit Change Enablement

- [ ] dauerhafte Lösung beschrieben
- [ ] Risiko bewertet
- [ ] betroffene Services und CIs bekannt
- [ ] Change-Typ geprüft
- [ ] Testbedarf beschrieben
- [ ] Rollback betrachtet
- [ ] Wartungsfenster geprüft
- [ ] Kommunikationsbedarf erkannt
- [ ] Change mit Problem Record verknüpft
- [ ] Wirksamkeit nach Change geprüft

Checkliste Zusammenspiel mit Knowledge Management

- [ ] Known Error dokumentiert
- [ ] Workaround beschrieben
- [ ] Zielgruppe des Artikels festgelegt
- [ ] Service Desk informiert
- [ ] Benutzerinformation bei Bedarf erstellt
- [ ] interne technische Hinweise ergänzt
- [ ] Eskalationskriterien beschrieben

- [ ] Artikel nach Change aktualisiert
- [ ] veraltete Inhalte entfernt
- [ ] Lessons Learned übernommen

Checkliste Zusammenspiel mit Configuration Management

- [ ] betroffene CIs erfasst
- [ ] Serviceabhängigkeiten geprüft
- [ ] Versionen dokumentiert
- [ ] Standorte berücksichtigt
- [ ] Lieferantenbezug geprüft
- [ ] letzte Changes berücksichtigt
- [ ] CMDB-Abweichungen erkannt
- [ ] notwendige Aktualisierungen veranlasst

Checkliste Abschluss eines Problems

- [ ] Ursache oder Risiko ausreichend verstanden
- [ ] Workaround dokumentiert oder als nicht verfügbar gekennzeichnet
- [ ] Known Error aktualisiert
- [ ] dauerhafte Lösung umgesetzt oder bewusst zurückgestellt
- [ ] Change-Ergebnis geprüft
- [ ] Knowledge Base aktualisiert
- [ ] zugehörige Incidents berücksichtigt
- [ ] Monitoring oder Runbooks angepasst
- [ ] Lessons Learned dokumentiert
- [ ] Rest Risiko bewertet
- [ ] Verantwortliche informiert

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker für Systemintegration stehen oft genau an der Schnittstelle zwischen Incident, Problem, Change und Knowledge.

Im Arbeitsalltag bedeutet das:

- Incidents sauber dokumentieren,

- wiederkehrende Muster erkennen,
- technische Ursachen nachvollziehbar beschreiben,
- Workarounds sicher formulieren,
- Knowledge-Artikel und Runbooks unterstützen,
- Changes fachlich vorbereiten,
- CIs und Abhängigkeiten berücksichtigen,
- Monitoring verbessern,
- und Lessons Learned in dauerhafte Verbesserungen überführen.

Gute technische Arbeit endet nicht mit der Wiederherstellung eines Services.

Sie hilft auch, zukünftige Incidents zu vermeiden oder schneller zu lösen.

Zusammenfassung

> Incident wird bearbeitet
> ↓
> Muster oder Ursache wird erkannt
> ↓
> Problem Record wird erstellt
> ↓
> Ursache, Risiko und Workaround werden untersucht
> ↓
> Known Error und Knowledge werden dokumentiert
> ↓
> dauerhafte Lösung wird vorbereitet
> ↓
> Change Enablement steuert Umsetzung
> ↓
> Wirksamkeit wird geprüft
> ↓
> Monitoring, Knowledge, Prozesse und Standards werden verbessert
> ↓
> zukünftige Incidents werden reduziert oder schneller lösbar

---

Merksätze

> Incident Management stellt den Service wieder her.

>

> Problem Management versteht Ursachen und reduziert Wiederholungen.

> Change Enablement sorgt für kontrollierte dauerhafte Lösungen.

> Knowledge Management macht Workarounds und Known Errors nutzbar.

> Configuration Management zeigt Abhängigkeiten und betroffene Komponenten.

> Continual Improvement sorgt dafür, dass Erkenntnisse nicht verloren gehen.

> Gute Zusammenarbeit verhindert, dass dieselben Fehler immer wieder neu analysiert werden.

Verwandte Seiten

- 4.1 Problem Management – Ziele, Begriffe und Abgrenzung
- 4.2 Ursachenanalyse
- 4.3 Workarounds und Known Errors
- 4.4 Trendanalyse und proaktives Problem Management
- 3.6 Erstdiagnose, Lösung und funktionale Eskalation
- 3.9 Kennzahlen, Qualität und Continual Improvement im Service Desk
- 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: Change Enablement

- PeopleCert – ITIL Practice Guide: Knowledge Management
- PeopleCert – ITIL Practice Guide: Service Configuration Management
- PeopleCert – ITIL Practice Guide: Continual Improvement
- ITIL Foundation – Version 5

Einordnung

Die dargestellten:

- Schnittstellen,
- Checklisten,
- Praxisbeispiele,
- Ablaufdarstellungen,
- Rollenhinweise,
- und Dokumentationshinweise

sind herstellerneutrale Praxisempfehlungen.

ITIL schreibt keine universelle:

- Schnittstellenmatrix,
- Übergabestruktur,
- Problem-Abschlussregel,
- Knowledge-Vorlage,
- Change-Verknüpfung,
- oder konkrete Toolintegration

für alle Organisationen vor.

Die konkrete Umsetzung muss an:

- Services,
- Risiken,
- Organisation,
- ITSM-Werkzeuge,
- Datenqualität,
- Supportmodell,
- Change-Modell,
- Knowledge-Struktur,
- Lieferanten,
- und verfügbare Fähigkeiten

angepasst werden.

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