6.5 Service Configuration Management im Zusammenspiel mit Incident-, Problem- und Change-Management

> **Kurz erklärt**
>
> Service Configuration Management liefert wichtige Informationen über Services, Configuration Items, Versionen, Verantwortlichkeiten und Abhängigkeiten.
>
> Diese Informationen werden besonders in Incident Management, Problem Management und Change Enablement benötigt.
>
> Ohne aktuelle Configuration-Daten werden Störungen langsamer analysiert, Ursachen schwerer erkannt und Changes riskanter bewertet.

Warum das Zusammenspiel wichtig ist

Configuration-Daten sind keine reine Dokumentation.

Sie unterstützen tägliche Entscheidungen im IT-Betrieb.

Beispiele:

- Welcher Service ist von einer Störung betroffen?
- Welche Systeme hängen von einer Datenbank ab?
- Wer ist Owner eines betroffenen Configuration Items?
- Welche Changes wurden zuletzt an einem System durchgeführt?
- Welche Version ist installiert?
- Welche Services nutzen ein bestimmtes Zertifikat?
- Welche Benutzergruppen sind betroffen?
- Welche Lieferanten müssen eingebunden werden?
- Welche Abhängigkeiten müssen vor einem Change geprüft werden?

Service Configuration Management liefert die Informationsbasis.

Incident, Problem und Change Management nutzen diese Informationen für Analyse, Bewertung und Steuerung.

Grundidee des Zusammenspiels

```
Service Configuration Management
        ↓
liefert Services, CIs, Beziehungen, Owner und Versionen
        ↓
Incident Management nutzt diese Daten für schnelle Diagnose
        ↓
Problem Management nutzt diese Daten für Ursachenanalyse
        ↓
Change Enablement nutzt diese Daten für Auswirkungs- und Risikobewertung
        ↓
Changes, Incidents und Problems liefern neue Erkenntnisse zurück
        ↓
Configuration-Daten werden aktualisiert und verbessert
```

Der Kreislauf funktioniert nur, wenn Configuration-Daten gepflegt und aktiv genutzt werden.

Zusammenspiel mit Incident Management

Incident Management benötigt Configuration-Daten, um Störungen schneller einzuordnen.

Wichtige Fragen:

- Welcher Service ist betroffen?
- Welche CIs gehören zum Service?
- Welche Abhängigkeiten bestehen?
- Wer ist zuständig?
- Welche Kritikalität besitzt der Service?
- Welche Changes wurden zuletzt durchgeführt?
- Gibt es Known Errors?
- Welche anderen Services könnten betroffen sein?
- Welche Dokumentation oder Runbooks sind verknüpft?

Je besser diese Informationen verfügbar sind, desto schneller kann der Incident bearbeitet werden.

Incident Management liefert Informationen zurück

Incidents können zeigen, dass Configuration-Daten falsch oder unvollständig sind.

Beispiele:

- falscher Owner im Ticket,
- fehlende Servicezuordnung,

- unbekannte Abhängigkeit,
- nicht erfasstes Zertifikat,
- veraltete Version,
- falscher CI-Status,
- fehlender Dokumentationslink,
- nicht dokumentierte Schnittstelle.

Diese Erkenntnisse sollten nicht im Incident verschwinden.

Sie sollten genutzt werden, um die CMDB oder das Configuration Management System zu verbessern.

Beispiel: Incident mit fehlender Beziehung

Situation

Benutzer melden, dass das Mitarbeiterportal nicht erreichbar ist.

Analyse

Der Webserver ist erreichbar.

Die Anwendung kann sich aber nicht mit der Datenbank verbinden.

Problem

In der CMDB war die Datenbankbeziehung nicht gepflegt.

Folge

Die Diagnose dauert länger.

Verbesserung

Die Beziehung zwischen Mitarbeiterportal, Anwendung, Datenbank, Storage und Backup wird nachgetragen.

Zusammenspiel mit Problem Management

Problem Management nutzt Configuration-Daten, um Ursachen und Muster zu erkennen.

Wichtige Fragen:

- Betreffen mehrere Incidents dasselbe CI?
- Gibt es eine gemeinsame Abhängigkeit?
- Ist eine bestimmte Version auffällig?

- Sind bestimmte Standorte oder Netzwerkkomponenten betroffen?
- Gibt es mehrere Services mit gleichem Zertifikat?
- Gibt es bekannte Lieferantenprobleme?
- Wurden ähnliche Problems bereits dokumentiert?
- Gibt es offene Known Errors?

Configuration-Daten helfen, einzelne Incidents in einen größeren Zusammenhang einzuordnen.

Problem Management liefert Informationen zurück

Problem Management kann Configuration-Daten verbessern.

Beispiele:

- neue CI-Beziehung wird erkannt,
- Known Error wird mit CI verknüpft,
- fehlerhafte Version wird dokumentiert,
- gemeinsame Abhängigkeit wird sichtbar,
- fehlender Owner wird erkannt,
- Zertifikat wird als eigenes CI aufgenommen,
- Schnittstelle wird nachträglich dokumentiert,
- Service Map wird ergänzt.

Eine Ursachenanalyse sollte deshalb auch prüfen, ob Configuration-Daten angepasst werden müssen.

Beispiel: Problem durch gemeinsame Abhängigkeit

Situation

Drei Anwendungen erzeugen ähnliche Fehlermeldungen.

Einzelbetrachtung

Jedes Team untersucht zunächst seine eigene Anwendung.

Configuration-Daten zeigen

- alle Anwendungen nutzen denselben Datenbankcluster,
- der Datenbankcluster nutzt denselben Storage,
- Storage zeigt erhöhte Latenz.

Ergebnis

Problem Management erkennt eine gemeinsame Ursache schneller.

Verbesserung

Die Serviceabhängigkeiten werden in der CMDB klarer gepflegt.

Zusammenspiel mit Change Enablement

Change Enablement benötigt Configuration-Daten für die Auswirkungs- und Risikobewertung.

Wichtige Fragen:

- Welche CIs werden geändert?
- Welche Services hängen davon ab?
- Welche Benutzer oder Standorte sind betroffen?
- Welche Owner müssen eingebunden werden?
- Welche Lieferanten sind beteiligt?
- Welche Versionen sind betroffen?
- Welche Sicherheitsrisiken entstehen?
- Welche anderen Changes betreffen dieselben CIs?
- Welche Tests sind erforderlich?
- Welche Dokumentation muss aktualisiert werden?

Ohne Configuration-Daten wird ein Change schnell zu niedrig bewertet.

Change Enablement liefert Informationen zurück

Changes verändern Configuration-Daten.

Beispiele:

- neue Version,
- neue Anwendung,
- neue Schnittstelle,
- neue Cloud-Ressource,
- geänderte Firewall-Regel,
- erneuertes Zertifikat,
- geänderte Datenbank,
- neuer Owner,
- außer Betrieb genommenes System,
- geänderte Monitoringregel.

Nach einem Change muss geprüft werden:

> “ Welche Configuration-Daten müssen aktualisiert werden?

Wenn Changes die CMDB nicht aktualisieren, veraltet sie automatisch.

Beispiel: Change mit unvollständiger Auswirkungsbewertung

Situation

Eine Firewall-Regel wird geändert.

Annahme

Nur eine kleine technische Anpassung.

Tatsächliche Auswirkung

Die Regel betrifft eine Schnittstelle zwischen Warenwirtschaft und Versand.

Nach dem Change können Aufträge nicht mehr übertragen werden.

Ursache

Die Beziehung zwischen Firewall-Regel, Schnittstelle und Geschäftsprozess war nicht dokumentiert.

Verbesserung

Schnittstellen und Firewall-Abhängigkeiten werden als relevante CIs und Beziehungen gepflegt.

Configuration-Daten im Change Request

Ein Change Request sollte relevante Configuration-Daten enthalten.

Dazu gehören:

- betroffene CIs,
- betroffene Services,
- Umgebung,
- Kritikalität,
- Owner,
- Supportgruppe,
- Version,
- Beziehungen,
- letzte Changes,
- Dokumentationslinks,
- bekannte Problems,
- Known Errors,
- Sicherheitsklassifizierung,
- und betroffene Lieferanten.

So wird aus einer technischen Änderung eine bewertbare Serviceänderung.

Zusammenspiel mit Release und Deployment Management

Release und Deployment Management benötigen Configuration-Daten für kontrollierte Bereitstellung.

Wichtige Fragen:

- Welche Version wird deployed?
- Welche Zielumgebung ist betroffen?
- Welche Komponenten gehören zum Release?
- Welche Abhängigkeiten bestehen?
- Welche Schnittstellen müssen funktionieren?
- Welche Services sind betroffen?
- Welche Knowledge-Artikel müssen aktualisiert werden?
- Welche CIs erhalten einen neuen Status?
- Welche CMDB-Daten ändern sich nach dem Deployment?

Nach einem Release müssen Versionen, Beziehungen und Dokumentation aktualisiert werden.

Beispiel: Release mit Versionsänderung

Situation

Eine neue Anwendungsversion wird ausgerollt.

Configuration Management muss aktualisieren

- Anwendungsversion,
- betroffene Server,
- Datenbankversion, falls geändert,
- Schnittstellen,
- Known Errors,
- Dokumentationslink,
- Service Map,
- Monitoringbeziehung.

Nutzen

Bei späteren Incidents ist sofort sichtbar, welche Version produktiv läuft.

Zusammenspiel mit Knowledge Management

Configuration-Daten und Knowledge Management ergänzen sich.

Die CMDB zeigt:

- welche CIs betroffen sind,
- welche Services abhängen,
- wer zuständig ist,
- welche Versionen genutzt werden,
- welche Beziehungen bestehen.

Knowledge Management liefert:

- Runbooks,
- Workarounds,
- Known Errors,
- Diagnoseanleitungen,
- Benutzerinformationen,
- Eskalationshinweise,
- Betriebsdokumentation.

Beide müssen verknüpft sein.

Eine CMDB ohne Dokumentationslinks ist weniger nützlich.

Eine Knowledge Base ohne Service- und CI-Bezug ist schwerer auffindbar.

Beispiel: Knowledge und CMDB

Situation

Ein VPN-Problem tritt wiederholt auf.

CMDB zeigt

- VPN-Service,
- VPN-Gateway,
- Clientversion,
- MFA-Abhängigkeit,
- Identity Provider,
- Supportgruppe.

Knowledge Base zeigt

- Known Error,
- Workaround,
- Prüfschritte,
- Eskalationsweg.

Nutzen

Der Service Desk kann neue Incidents schneller erkennen und bearbeiten.

Zusammenspiel mit Information Security Management

Information Security Management benötigt Configuration-Daten für Risikobewertung und Schutzmaßnahmen.

Wichtige Fragen:

- Welche Systeme sind kritisch?
- Welche CIs sind extern erreichbar?
- Welche Systeme verarbeiten vertrauliche Daten?
- Welche Versionen sind verwundbar?
- Welche Zertifikate laufen bald ab?
- Welche Benutzergruppen haben Zugriff?
- Welche Schnittstellen übertragen sensible Daten?
- Welche Lieferanten haben Zugriff?
- Welche Systeme sind nicht inventarisiert?

Ohne Configuration-Daten ist Sicherheitsarbeit unvollständig.

Beispiel: Sicherheitslücke

Situation

Eine kritische Schwachstelle betrifft eine bestimmte Softwareversion.

Mit guter CMDB

Es ist erkennbar:

- welche Server diese Version nutzen,
- welche Services betroffen sind,
- welche Systeme produktiv sind,
- welche Systeme extern erreichbar sind,
- wer Owner ist,
- welche Changes zur Behebung nötig sind.

Nutzen

Patchpriorisierung wird schneller und nachvollziehbarer.

Zusammenspiel mit IT Asset Management

IT Asset Management und Service Configuration Management überschneiden sich.

IT Asset Management liefert Informationen über:

- Besitz,
- Beschaffung,
- Kosten,
- Lizenzen,
- Verträge,
- Lebenszyklus,
- Garantie,
- Lieferanten,
- wirtschaftliche Nutzung.

Service Configuration Management ergänzt:

- Servicebezug,
- Abhängigkeiten,
- Betriebsrelevanz,
- Changes,
- Incidents,
- Problems,
- technische Beziehungen.

Ein Server kann gleichzeitig Asset und CI sein.

Beide Sichten sollten zusammenpassen.

Zusammenspiel mit Service Level Management

Service Level Management benötigt Configuration-Daten, um Service Levels realistisch zu bewerten.

Wichtige Fragen:

- Welche Komponenten unterstützen einen kritischen Service?
- Welche CIs benötigen hohe Verfügbarkeit?
- Welche Lieferanten beeinflussen Service Levels?
- Welche Abhängigkeiten können SLA-Verletzungen verursachen?
- Welche Changes gefährden Service Levels?
- Welche Incidents betreffen kritische CIs?

Service Levels sind schwer steuerbar, wenn die Serviceabhängigkeiten unbekannt sind.

Zusammenspiel mit Monitoring

Monitoring und Configuration Management ergänzen sich.

Monitoring zeigt:

- aktueller Zustand,
- Alarme,
- Fehler,
- Performance,
- Kapazität,
- Verfügbarkeit.

Configuration Management zeigt:

- Servicebezug,
- Owner,
- Kritikalität,
- Beziehungen,
- Version,
- Dokumentationslink.

Beispiel:

Monitoring meldet einen Serverausfall.

Die CMDB zeigt, dass daran ein geschäftskritischer Service hängt.

Dadurch kann der Incident korrekt priorisiert und kommuniziert werden.

Zusammenspiel mit Continual Improvement

Configuration Management liefert viele Verbesserungsansätze.

Beispiele:

- CIs ohne Owner,
- Services ohne Abhängigkeiten,
- Changes ohne CI-Verknüpfung,
- Incidents ohne Servicezuordnung,
- veraltete Versionen,
- Zertifikate ohne Ablaufdatum,
- Cloud-Ressourcen ohne Tags,
- kritische CIs ohne Monitoring,
- Dubletten,
- veraltete Dokumentationslinks.

Diese Erkenntnisse sollten in Continual Improvement einfließen.

Typischer Informationsfluss

```
Incident erkennt Datenfehler
        ↓
Datenfehler wird dokumentiert
        ↓
Configuration Management prüft und korrigiert
        ↓
Change aktualisiert CI oder Beziehung
        ↓
Problem Management erkennt Muster
        ↓
Knowledge Management ergänzt Workaround
        ↓
Reporting zeigt Datenqualität
        ↓
Continual Improvement verbessert Prozess und Pflege
```

Configuration Management ist dadurch Teil eines Lernkreislaufs.

Wichtige Schnittstelleninformationen

Für das Zusammenspiel mit anderen Practices sind besonders wichtig:

- Servicezuordnung,
- CI-Beziehungen,
- Owner,
- Supportgruppe,
- Kritikalität,
- Umgebung,
- Status,
- Version,
- letzte Changes,
- Known Errors,
- Dokumentationslinks,
- Sicherheitsklassifizierung,
- Lieferantenbezug,
- Monitoringbezug,
- Backupbezug.

Diese Informationen sollten nicht nur vorhanden sein, sondern auch aktiv genutzt werden.

Praxisbeispiel: Major Incident

Situation

Ein zentraler Service fällt aus.

Incident Management benötigt

- betroffener Service,
- abhängige CIs,
- Owner,
- Supportgruppen,
- Kritikalität,
- Kommunikationskontakte,
- letzte Changes,
- Known Errors.

Configuration Management liefert

- Service Map,
- CI-Beziehungen,
- betroffene Komponenten,
- Dokumentationslinks.

Nachbereitung

- fehlende Beziehungen ergänzen,
- falsche Owner korrigieren,
- Service Map aktualisieren,
- Monitoringbeziehungen prüfen.

Praxisbeispiel: Wiederkehrende Druckerprobleme

Situation

Ein Standort meldet regelmäßig Druckprobleme.

Incident-Daten zeigen

- mehrere Tickets,
- gleicher Standort,
- unterschiedliche Benutzer.

Configuration-Daten zeigen

- gleicher Druckserver,

- gleicher Treiber,
- gleiche Standort-Firewall,
- gleiche Fachanwendung.

Problem Management erkennt

Der Fehler hängt mit einem alten Treiber und einer bestimmten Anwendung zusammen.

Verbesserung

Treiber wird über Change Enablement aktualisiert.

Knowledge-Artikel und CMDB werden angepasst.

Praxisbeispiel: Zertifikatsmanagement

Situation

Ein Zertifikat läuft ab und verursacht einen Incident.

Problem Management erkennt

- Zertifikat war nicht als CI erfasst,
- kein Owner dokumentiert,
- kein Ablaufmonitoring,
- kein Erneuerungs-Runbook.

Change Enablement setzt um

- Zertifikat erneuern,
- Monitoring ergänzen,
- Owner festlegen,
- Runbook erstellen.

Configuration Management aktualisiert

- Zertifikat als CI,
- Ablaufdatum,
- Servicebeziehung,
- Owner,
- Monitoringbeziehung,
- Dokumentationslink.

Praxisbeispiel: Cloud-Ressource ohne Owner

Situation

Eine Cloud-Ressource erzeugt Kosten und ist sicherheitsrelevant.

Problem

Kein Owner, kein Service-Tag, keine Dokumentation.

Risiko

- niemand fühlt sich verantwortlich,
- Sicherheitsbewertung unklar,
- Außerbetriebnahme riskant,
- Kosten nicht zuordenbar.

Verbesserung

- Tag-Regeln einführen,
- Owner verpflichtend machen,
- Cloud-Discovery anbinden,
- Ressourcen ohne Tags regelmäßig berichten,
- Change-Prozess anpassen.

Typische Fehler im Zusammenspiel

Fehler 1

CMDB wird gepflegt, aber im Incident Management nicht genutzt.

Fehler 2

Incidents zeigen Datenfehler, aber niemand korrigiert sie.

Fehler 3

Problem Management erkennt neue Abhängigkeiten, aber sie werden nicht dokumentiert.

Fehler 4

Changes ändern CIs, ohne die CMDB zu aktualisieren.

Fehler 5

Release und Deployment aktualisieren Versionen nicht.

Fehler 6

Knowledge-Artikel sind nicht mit Services oder CIs verknüpft.

Fehler 7

Security bewertet Risiken ohne aktuelle Configuration-Daten.

Fehler 8

Monitoring-Alarme enthalten keine Serviceinformationen.

Fehler 9

Asset-Daten und CI-Daten widersprechen sich.

Fehler 10

Owner und Supportgruppen sind veraltet.

Fehler 11

Configuration-Daten werden nur für Audits gepflegt, nicht für den Betrieb.

Fehler 12

Datenqualitätsprobleme werden gemessen, aber nicht verbessert.

Checkliste Zusammenspiel mit Incident Management

- ☐ Incident enthält betroffenen Service
- ☐ relevante CIs sind verknüpft
- ☐ Kritikalität ist sichtbar
- ☐ Owner und Supportgruppe sind erkennbar
- ☐ letzte Changes sind prüfbar
- ☐ Known Errors sind auffindbar
- ☐ Dokumentationslinks sind vorhanden
- ☐ Datenfehler können gemeldet werden
- ☐ fehlende Beziehungen werden nachgetragen

- [ ] wiederkehrende CI-bezogene Incidents werden ausgewertet

---

Checkliste Zusammenspiel mit Problem Management

- [ ] Problems sind mit relevanten CIs verknüpft
- [ ] gemeinsame Abhängigkeiten werden geprüft
- [ ] betroffene Versionen werden dokumentiert
- [ ] Known Errors werden mit CIs verknüpft
- [ ] neue Beziehungen aus RCA werden ergänzt
- [ ] fehlende Owner werden korrigiert
- [ ] Service Maps werden nach Analyse aktualisiert
- [ ] Workarounds werden mit Knowledge verknüpft
- [ ] Problem-Erkenntnisse verbessern Configuration-Daten

Checkliste Zusammenspiel mit Change Enablement

- [ ] Change Request enthält betroffene CIs
- [ ] abhängige Services wurden geprüft
- [ ] Kritikalität wurde berücksichtigt
- [ ] Owner wurden einbezogen
- [ ] Sicherheitsklassifizierung wurde geprüft
- [ ] parallele Changes an gleichen CIs wurden geprüft
- [ ] CMDB-Aktualisierung ist Teil des Change-Abschlusses
- [ ] neue CIs werden erstellt
- [ ] entfernte CIs werden archiviert
- [ ] geänderte Beziehungen werden aktualisiert

Checkliste Zusammenspiel mit Release und Deployment

- [ ] Release enthält betroffene CIs
- [ ] Zielumgebung ist korrekt dokumentiert
- [ ] Versionen werden nach Deployment aktualisiert
- [ ] Schnittstellen werden geprüft
- [ ] Known Errors werden aktualisiert
- [ ] Service Desk erhält relevante CI-Informationen

- [ ] Dokumentationslinks werden aktualisiert
- [ ] Monitoringbeziehungen werden geprüft
- [ ] CMDB-Abschluss ist Teil der Nachbereitung

Checkliste Zusammenspiel mit Security

- [ ] kritische CIs sind gekennzeichnet
- [ ] externe Erreichbarkeit ist dokumentiert
- [ ] Datenklassifizierung ist gepflegt
- [ ] Zertifikate sind als CIs erfasst
- [ ] Ablaufdaten werden überwacht
- [ ] privilegierte Gruppen sind dokumentiert
- [ ] veraltete Versionen sind auswertbar
- [ ] Schwachstellen können Services zugeordnet werden
- [ ] Lieferantenbezug ist sichtbar
- [ ] Sicherheitsrelevante Changes nutzen CI-Daten

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker arbeiten täglich an der Schnittstelle zwischen Technik, Support und Betrieb.

Für sie bedeutet das:

- bei Incidents betroffene CIs korrekt erfassen,
- bei Problems Abhängigkeiten erkennen,
- bei Changes betroffene Services prüfen,
- nach Änderungen CMDB-Daten aktualisieren,
- Versionen und Konfigurationen nachvollziehbar dokumentieren,
- falsche Owner oder Beziehungen melden,
- Knowledge-Artikel mit Services oder CIs verknüpfen,
- und technische Beobachtungen in nutzbare Betriebsinformationen überführen.

Gute Configuration-Daten entstehen nicht nur durch ein Tool.

Sie entstehen durch konsequente Nutzung im Alltag.

Zusammenfassung

> Configuration-Daten bereitstellen
> ↓
> Incident Management nutzt sie für Diagnose und Priorisierung
> ↓
> Problem Management nutzt sie für Muster und Ursachen
> ↓
> Change Enablement nutzt sie für Risiko- und Auswirkungsbewertung
> ↓
> Release und Deployment aktualisieren Versionen und Beziehungen
> ↓
> Knowledge Management verknüpft Workarounds und Runbooks
> ↓
> Security nutzt Daten für Risiko- und Schwachstellenbewertung
> ↓
> Datenfehler und neue Erkenntnisse fließen zurück
> ↓
> CMDB, Service Maps und Dokumentation werden verbessert

Merksätze

> Configuration-Daten sind nur wertvoll, wenn sie im Betrieb genutzt werden.

> Incident Management braucht Service- und CI-Bezug für schnelle Diagnose.

> Problem Management braucht Beziehungen, um gemeinsame Ursachen zu erkennen.

> Change Enablement braucht Abhängigkeiten für realistische Risikobewertung.

> Jeder relevante Change sollte Configuration-Daten aktualisieren.

>

> Knowledge ohne CI-Bezug ist schwerer auffindbar.

> Monitoring ohne Servicebezug erschwert Priorisierung.

> Gute Configuration-Daten verbessern Support, Stabilität, Sicherheit und Veränderungsfähigkeit.

Verwandte Seiten

- 6.1 Service Configuration Management – Ziele, Begriffe und Grundlagen
- 6.2 Configuration Items und Beziehungen
- 6.3 Configuration Management Database
- 6.4 Discovery, Pflege und Datenqualität
- Incident Management
- Problem Management
- Change Enablement
- Release Management
- Knowledge Management
- Information Security Management
- IT Asset Management
- Continual Improvement

Quellen und Versionsstand

Offizielle Grundlagen

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

Einordnung

Die dargestellten:

- Schnittstellen,
- Informationsflüsse,
- Praxisbeispiele,
- Checklisten,
- Rollenhinweise,
- und Datenqualitätsbezüge

sind herstellerneutrale Praxisempfehlungen.

ITIL schreibt keine universelle:

- Schnittstellenmatrix,
- CMDB-Prozessintegration,
- Pflichtverknüpfung,
- Service-Map-Struktur,
- Reporting-Form,
- oder Toolintegration

für alle Organisationen vor.

Die konkrete Umsetzung muss an:

- Services,
- Risiken,
- Organisationsgröße,
- Supportmodell,
- Change-Modell,
- Toollandschaft,
- Datenqualität,
- Sicherheitsanforderungen,
- Cloud-Nutzung,
- und verfügbare Fähigkeiten

angepasst werden.

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