13. Change Enablement

13.1 Ziele und Grundlagen des Change Enablement

> **Kurz erklärt**
>
> Änderungen an IT-Systemen gehören zum Alltag jeder Organisation.
>
> Neue Software wird installiert, Server aktualisiert, Sicherheitslücken geschlossen oder Cloud-Dienste erweitert.
>
> Ziel des **Change Enablement** ist es, diese Änderungen kontrolliert, nachvollziehbar und mit möglichst geringem Risiko umzusetzen.
>
> Dabei sollen notwendige Veränderungen ermöglicht werden, ohne die Stabilität der IT-Services unnötig zu gefährden.

Was ist Change Enablement?

Change Enablement ist eine ITIL Practice zur Steuerung und Unterstützung von Änderungen an Services und Configuration Items.

Sie sorgt dafür,

dass Änderungen:

- geplant,
- bewertet,
- genehmigt,
- umgesetzt,
- überprüft

und dokumentiert werden.

Warum heißt es nicht mehr Change Management?

Frühere ITIL-Versionen verwendeten den Begriff:

> **Change Management**

Seit ITIL 4 lautet die offizielle Bezeichnung:

> **Change Enablement**

Der neue Name verdeutlicht,

dass Änderungen nicht verhindert,

sondern sicher und effizient ermöglicht werden sollen.

Warum Änderungen notwendig sind

Ohne Änderungen wäre eine moderne IT nicht dauerhaft funktionsfähig.

Typische Gründe:

- Sicherheitsupdates,
- neue Hardware,
- neue Software,
- gesetzliche Anforderungen,
- Fehlerbehebungen,
- Leistungsverbesserungen,
- Cloud-Migrationen,
- neue Geschäftsanforderungen.

Veränderungen gehören zum normalen IT-Betrieb.

Ziele des Change Enablement

Die wichtigsten Ziele sind:

- Risiken reduzieren,
- erfolgreiche Änderungen ermöglichen,
- Serviceunterbrechungen vermeiden,
- Auswirkungen bewerten,
- Verantwortlichkeiten festlegen,
- Änderungen dokumentieren,
- Zusammenarbeit verbessern,
- kontinuierliche Verbesserungen unterstützen.

Was ist ein Change?

Ein Change ist jede geplante Änderung,

die Auswirkungen auf Services oder Configuration Items haben kann.

Beispiele:

- Betriebssystem aktualisieren,
- Firewallregel ändern,
- Benutzerverwaltung erweitern,
- Zertifikat erneuern,
- Server austauschen,
- Netzwerk erweitern,
- Cloud-Service aktivieren,
- Software installieren.

Nicht jede technische Tätigkeit ist automatisch ein Change.

Change ist nicht gleich Incident

Diese Begriffe werden häufig verwechselt.

Incident	Change
ungeplante Störung	geplante Änderung
Ziel: Service wiederherstellen	Ziel: Service verbessern oder anpassen
häufig Zeitdruck	normalerweise geplant
reaktive Tätigkeit	proaktive Tätigkeit

Ein Incident kann später einen Change erforderlich machen.

Beispiel

Ein Server fällt regelmäßig aus.

Incident Management stellt den Betrieb mehrfach wieder her.

Problem Management findet die Ursache.

Zur dauerhaften Lösung wird:

- zusätzlicher Arbeitsspeicher eingebaut.

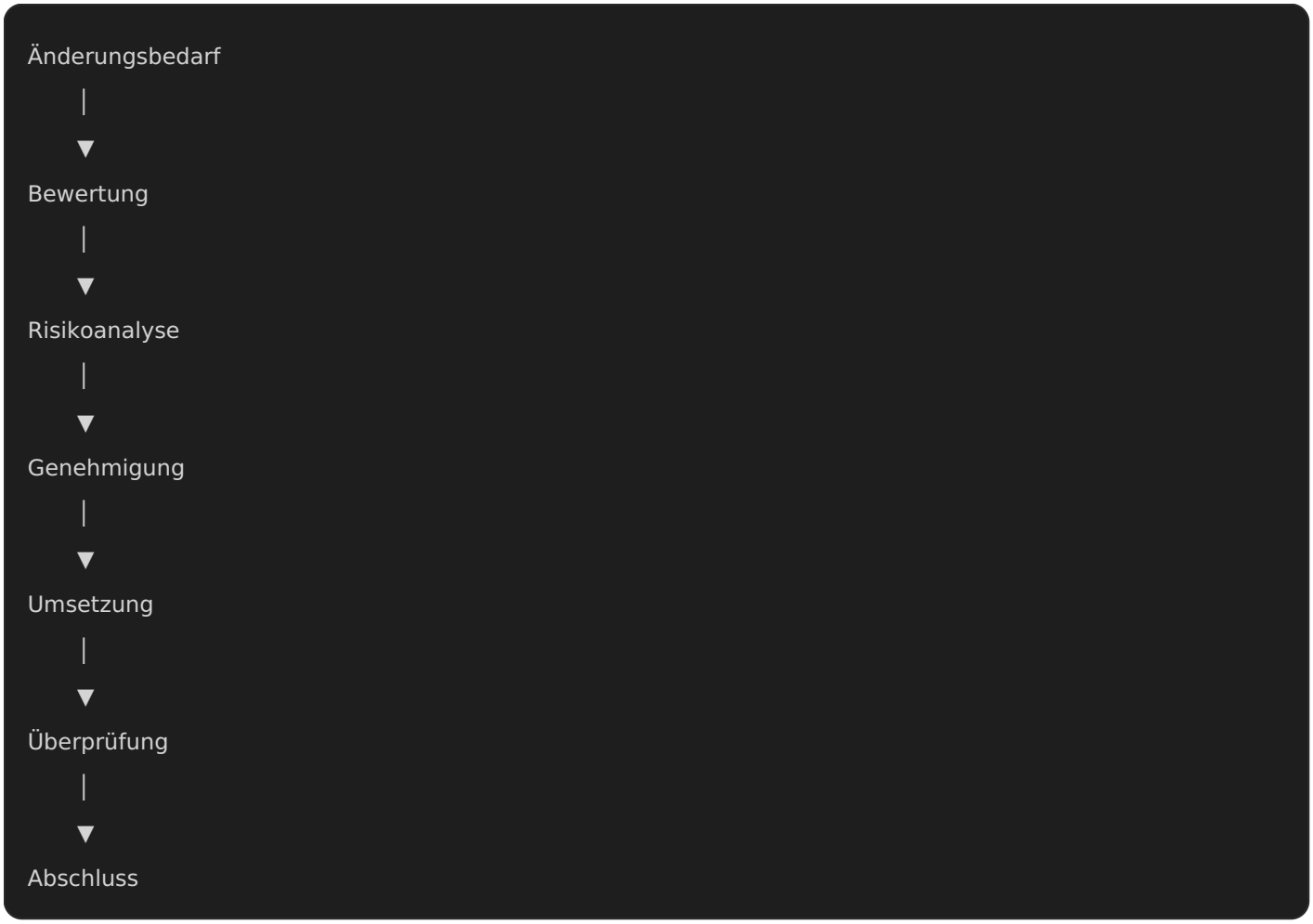
Diese geplante Änderung ist ein Change.

Grundprinzip des Change Enablement

Jede Änderung sollte beantwortete Fragen besitzen:

- Warum ist die Änderung notwendig?
- Welche Systeme sind betroffen?
- Welche Risiken bestehen?
- Wer genehmigt die Änderung?
- Wann wird sie durchgeführt?
- Wie wird sie getestet?
- Wie erfolgt ein Rollback?
- Wer informiert die Benutzer?

Typischer Ablauf



Nicht jeder Change durchläuft alle Schritte mit demselben Aufwand.

Warum Risiken bewertet werden

Jede Änderung kann unbeabsichtigte Auswirkungen haben.

Beispiele:

- Server startet nicht mehr,
- Anwendung funktioniert nicht,
- Netzwerkverbindung fällt aus,
- Benutzer können sich nicht anmelden,
- Daten gehen verloren.

Deshalb bewertet Change Enablement Risiken bereits vor der Umsetzung.

Nutzen strukturierter Changes

Ein geregelter Change-Prozess sorgt unter anderem für:

- weniger Ausfälle,
- bessere Planbarkeit,
- höhere Servicequalität,
- nachvollziehbare Entscheidungen,
- geringere Risiken,
- bessere Dokumentation.

Dadurch steigt die Stabilität der IT.

Nicht jede Änderung ist gleich kritisch

Beispiele:

Geringes Risiko:

- Druckertreiber aktualisieren.

Hohes Risiko:

- Active Directory migrieren.

Je größer die Auswirkungen,

desto sorgfältiger erfolgt Planung und Bewertung.

Verantwortlichkeiten

An einem Change können verschiedene Rollen beteiligt sein.

Beispiele:

- Antragsteller,
- Change Manager,

- technische Spezialisten,
- Service Owner,
- Information Security,
- Management.

Nicht jede Organisation verwendet alle Rollen.

Dokumentation

Jeder Change sollte nachvollziehbar dokumentiert werden.

Typische Inhalte:

- Ziel,
- Umfang,
- Zeitpunkt,
- Verantwortliche,
- Risiko,
- Genehmigung,
- Testergebnisse,
- Rollback,
- Abschlussbewertung.

Eine gute Dokumentation erleichtert spätere Analysen.

Praxisbeispiel

Ein Unternehmen möchte den VPN-Server aktualisieren.

Vor der Umsetzung werden:

- Risiken bewertet,
- Wartungsfenster festgelegt,
- Benutzer informiert,
- Backup erstellt,
- Rollback vorbereitet.

Nach erfolgreicher Aktualisierung bestätigt das Monitoring,

dass der Service wieder ordnungsgemäß funktioniert.

Typische Fehler

Fehler 1

Änderungen erfolgen ohne Planung.

Fehler 2

Risiken werden nicht bewertet.

Fehler 3

Es existiert kein Rollback.

Fehler 4

Benutzer werden nicht informiert.

Fehler 5

Änderungen werden nicht dokumentiert.

Fehler 6

Monitoring prüft das Ergebnis nicht.

Fehler 7

Configuration-Daten werden nicht aktualisiert.

Fehler 8

Changes werden direkt in der Produktion getestet.

Checkliste Change Enablement

- [ ] Ziel definiert
- [ ] Auswirkungen bewertet
- [ ] Risiken analysiert
- [ ] Genehmigung vorhanden
- [ ] Wartungsfenster geplant
- [ ] Rollback vorbereitet
- [ ] Monitoring berücksichtigt

- [ ] Dokumentation vollständig

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker setzen täglich Changes um.

Typische Aufgaben:

- Updates installieren,
- Server migrieren,
- Netzwerkänderungen durchführen,
- Firewallregeln anpassen,
- Dokumentationen aktualisieren,
- Monitoring prüfen,
- Rollback vorbereiten.

Ein strukturierter Change-Prozess reduziert Ausfälle und erhöht die Stabilität der IT.

Zusammenfassung

> Änderungsbedarf erkennen
>
> ↓
>
> Risiken bewerten
>
> ↓
>
> Genehmigung einholen
>
> ↓
>
> Änderung umsetzen
>
> ↓
>
> Ergebnis überprüfen
>
> ↓
>
> Dokumentieren und abschließen

Merksätze

> Change Enablement ermöglicht Veränderungen – es verhindert sie nicht.

> Jede Änderung sollte geplant und bewertet werden.

> Je höher das Risiko, desto sorgfältiger die Vorbereitung.

> Ein Rollback gehört zu jeder kritischen Änderung.

> Gute Dokumentation unterstützt zukünftige Changes.

Verwandte Seiten

- 13.2 Change-Typen und Risikobewertung
- 13.3 Genehmigungen, CAB und Change-Kalender
- 13.4 Umsetzung, Tests und Rollback
- 13.5 Zusammenspiel mit Incident, Problem, Service Configuration Management und Monitoring
- Incident Management
- Problem Management
- Service Configuration Management
- Monitoring and Event Management

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Change Enablement
- ITIL Foundation – Version 5

Einordnung

Die dargestellten Grundlagen orientieren sich an der ITIL Practice „Change Enablement“. ITIL definiert Grundprinzipien für die kontrollierte Durchführung von Änderungen, schreibt jedoch keinen starren Prozess vor. Die konkrete Umsetzung richtet sich nach Größe, Risiken und Anforderungen der jeweiligen Organisation.

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

13.2 Change-Typen und Risikobewertung

> **Kurz erklärt**
>
> Nicht jede Änderung besitzt dasselbe Risiko oder denselben Aufwand.
>
> Deshalb unterscheidet ITIL verschiedene Change-Typen.
>
> Je nach Risiko, Auswirkungen und Dringlichkeit unterscheiden sich Planung, Genehmigung und Durchführung.
>
> Ziel ist es, **einfache Änderungen möglichst effizient und risikoreiche Änderungen besonders sorgfältig umzusetzen.**

Warum verschiedene Change-Typen?

Eine kleine Änderung,

wie das Aktualisieren einer Druckersoftware,

benötigt deutlich weniger Planung als die Migration eines Active Directory.

Würden beide Änderungen denselben Prozess durchlaufen,

wäre der Aufwand unnötig hoch.

Deshalb unterscheidet ITIL verschiedene Change-Typen.

Die drei Change-Typen nach ITIL

ITIL Version 5 unterscheidet grundsätzlich:

Change-Typ	Beschreibung
Standard Change	geringes Risiko, vorab genehmigt
Normal Change	individuelle Bewertung und Genehmigung erforderlich
Emergency Change	schnelle Umsetzung wegen dringender Situation

Diese Einteilung richtet sich nach Risiko und Auswirkungen,

nicht nach der technischen Komplexität.

Standard Change

Ein Standard Change ist:

- bekannt,
- dokumentiert,
- risikoarm,
- getestet,
- bereits genehmigt.

Für jede Durchführung ist normalerweise keine neue Genehmigung erforderlich.

Beispiele für Standard Changes

Mögliche Standard Changes:

- Druckertreiber installieren,
- Benutzerkonto anlegen,
- Standardsoftware installieren,
- Zertifikat erneuern,
- Gruppenmitgliedschaft ändern,
- Standard-Firewallregel aktivieren,
- VM nach Vorlage bereitstellen.

Diese Änderungen folgen meist einem festen Ablauf.

Eigenschaften eines Standard Change

Ein Standard Change besitzt:

- klaren Ablauf,
- bekannte Risiken,
- dokumentierte Arbeitsschritte,
- definierte Verantwortlichkeiten,
- erprobte Durchführung,
- festgelegte Rollback-Möglichkeiten.

Dadurch kann er effizient umgesetzt werden.

Normal Change

Ein Normal Change ist die häufigste Form einer Änderung.

Er wird individuell bewertet.

Dabei werden unter anderem betrachtet:

- Risiken,
- Auswirkungen,
- Aufwand,
- Ressourcen,
- Wartungsfenster,
- Genehmigungen.

Beispiele für Normal Changes

Typische Beispiele:

- Servermigration,
- Firewall-Konfiguration ändern,
- neue Anwendung einführen,
- Datenbank aktualisieren,
- Netzwerk erweitern,
- Active Directory ändern,
- Cluster erweitern.

Diese Änderungen benötigen meist eine individuelle Planung.

Emergency Change

Ein Emergency Change dient dazu,

eine akute Gefahr oder Störung schnell zu beseitigen.

Dabei wird der normale Genehmigungsprozess verkürzt,

nicht jedoch vollständig ausgelassen.

Beispiele für Emergency Changes

Typische Situationen:

- kritische Sicherheitslücke,
- Ransomware-Angriff,
- Zertifikat abgelaufen,
- Produktionssystem ausgefallen,
- kritischer Softwarefehler,
- schwerwiegender Netzwerkausfall.

Hier steht die schnelle Wiederherstellung des Betriebs im Vordergrund.

Emergency bedeutet nicht ungeplant

Auch Emergency Changes sollten soweit möglich:

- dokumentiert,
- bewertet,
- getestet,
- nachbereitet

werden.

Lediglich der Zeitdruck ist deutlich höher.

Vergleich der Change-Typen

Merkmal	Standard	Normal	Emergency
Risiko	gering	unterschiedlich	häufig hoch
Genehmigung	vorab	individuell	beschleunigt
Planung	standardisiert	individuell	verkürzt
Dokumentation	erforderlich	erforderlich	ebenfalls erforderlich

---

Risikobewertung

Vor jeder Änderung sollte bewertet werden,

welche Risiken bestehen.

Typische Fragen:

- Welche Services sind betroffen?
- Welche Benutzer sind betroffen?
- Welche Ausfallzeit entsteht?
- Gibt es Redundanzen?
- Existiert ein Rollback?
- Wurde ausreichend getestet?

Die Risikobewertung unterstützt fundierte Entscheidungen.

Risikofaktoren

Beispiele für Risikofaktoren:

- produktive Systeme,
- geschäftskritische Services,
- viele Benutzer betroffen,
- komplexe Infrastruktur,
- neue Technologien,
- fehlende Erfahrung,
- hoher Zeitdruck,
- externe Abhängigkeiten.

Je mehr Risikofaktoren vorliegen,

desto sorgfältiger sollte geplant werden.

Business Impact

Neben technischen Risiken wird auch der geschäftliche Einfluss bewertet.

Beispiele:

Gering:

- Testsystem.

Mittel:

- internes Intranet.

Hoch:

- ERP-System,
- Produktionssteuerung,
- Online-Shop.

Business Impact und technisches Risiko sind nicht immer identisch.

Wahrscheinlichkeit und Auswirkung

Risiken werden häufig anhand zweier Kriterien bewertet:

- Eintrittswahrscheinlichkeit,
- Auswirkung.

Beispiel:

Wahrscheinlichkeit	Auswirkung	Risiko

gering	gering	niedrig
hoch	gering	mittel
gering	hoch	mittel
hoch	hoch	hoch

Diese Bewertung unterstützt die Entscheidung über notwendige Maßnahmen.

Risikomatrix

Eine vereinfachte Risikomatrix:

```
            Auswirkung
          Niedrig  Mittel  Hoch
Niedrig      □□      □□     □□
Mittel       □□      □□     □□
Hoch         □□      □□     □□
```

Je höher das Risiko,

desto umfangreicher sollten Planung und Genehmigung sein.

Risikominimierung

Risiken lassen sich häufig reduzieren.

Beispiele:

- Tests durchführen,
- Wartungsfenster nutzen,
- Backup erstellen,
- Rollback vorbereiten,
- Pilotbetrieb,
- Redundanzen verwenden,
- Benutzer informieren.

Ziel ist nicht,

jedes Risiko vollständig auszuschließen,

sondern es auf ein akzeptables Maß zu reduzieren.

Rest-Risiko

Auch nach sorgfältiger Planung bleibt häufig ein Restrisiko bestehen.

Beispiel:

Ein Betriebssystem-Update wurde erfolgreich getestet.

Trotzdem kann es in der Produktivumgebung zu unerwarteten Problemen kommen.

Deshalb gehören Rollback und Monitoring zu jeder kritischen Änderung.

Praxisbeispiel

Ein Unternehmen plant ein Firmware-Update für zentrale Switches.

Die Bewertung ergibt:

- hoher Business Impact,
- mittlere Eintrittswahrscheinlichkeit,
- vorhandene Redundanz,
- getesteter Rollback.

Der Change wird als Normal Change durchgeführt.

Das Wartungsfenster wird nachts geplant,

Monitoring überwacht die Umsetzung

und bei Problemen steht ein Rollback bereit.

Typische Fehler

Fehler 1

Alle Änderungen werden gleich behandelt.

Fehler 2

Risiken werden nicht dokumentiert.

Fehler 3

Business Impact wird unterschätzt.

Fehler 4

Emergency Changes werden nicht nachbereitet.

Fehler 5

Standard Changes werden nie überprüft.

Fehler 6

Es existiert kein Rollback.

Fehler 7

Tests fehlen.

Fehler 8

Benutzer werden nicht informiert.

Fehler 9

Monitoring überwacht den Change nicht.

Fehler 10

Risikobewertung erfolgt nur technisch,

nicht geschäftlich.

Checkliste Risikobewertung

- [ ] Change-Typ bestimmt
- [ ] Auswirkungen bewertet
- [ ] Business Impact geprüft
- [ ] Risiken dokumentiert
- [ ] Tests durchgeführt
- [ ] Rollback vorbereitet
- [ ] Wartungsfenster geplant

- [ ] Monitoring berücksichtigt

Checkliste Change-Typen

- [ ] Standard Change geeignet?
- [ ] Normal Change erforderlich?
- [ ] Emergency Change begründet?
- [ ] Genehmigung vorhanden
- [ ] Dokumentation aktuell
- [ ] Verantwortlichkeiten bekannt
- [ ] Nachbereitung geplant

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker bewerten und begleiten regelmäßig Änderungen.

Typische Aufgaben:

- Risiken einschätzen,
- Auswirkungen analysieren,
- Tests durchführen,
- Rollback vorbereiten,
- Monitoring überwachen,
- Dokumentationen aktualisieren.

Eine realistische Risikobewertung gehört zu den wichtigsten Voraussetzungen erfolgreicher Changes.

Zusammenfassung

> Änderungsbedarf erkennen
>
> ↓
>
> Change-Typ bestimmen
>
> ↓
>
> Risiken bewerten

> ↓
>
> Business Impact analysieren
>
> ↓
>
> Maßnahmen planen
>
> ↓
>
> Genehmigung vorbereiten

Merksätze

> Nicht jede Änderung benötigt denselben Aufwand.

> Standard Changes sind vorab genehmigt und risikoarm.

> Normal Changes werden individuell bewertet.

> Emergency Changes beschleunigen den Prozess – sie ersetzen ihn nicht.

> Jede Risikobewertung sollte technische und geschäftliche Auswirkungen berücksichtigen.

Verwandte Seiten

- 13.1 Ziele und Grundlagen des Change Enablement
- 13.3 Genehmigungen, CAB und Change-Kalender
- 13.4 Umsetzung, Tests und Rollback
- 13.5 Zusammenspiel mit Incident, Problem, Service Configuration Management und Monitoring
- Incident Management

- Problem Management
- Service Configuration Management

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Change Enablement
- ITIL Foundation – Version 5

Einordnung

Die dargestellten Change-Typen und Verfahren zur Risikobewertung entsprechen den Empfehlungen der ITIL Practice „Change Enablement“. ITIL schreibt keine feste Risikomatrix oder konkrete Bewertungsmethode vor. Organisationen definieren diese entsprechend ihrer Geschäftsanforderungen und ihrer Risikobereitschaft.

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

13.3 Genehmigungen, Change Authority, CAB und Change-Kalender

> **Kurz erklärt**
>
> Nicht jede Änderung darf sofort umgesetzt werden.
>
> Abhängig vom Risiko, den Auswirkungen und der Bedeutung eines Changes müssen geeignete Personen oder Gremien über dessen Durchführung entscheiden.
>
> ITIL verwendet hierfür den Begriff **Change Authority**.
>
> Zusätzlich unterstützen ein **Change Advisory Board (CAB)** sowie ein **Change-Kalender** die koordinierte Planung und Durchführung von Änderungen.

Warum Genehmigungen notwendig sind

Änderungen können erhebliche Auswirkungen auf IT-Services haben.

Beispiele:

- Ausfall geschäftskritischer Systeme,
- Sicherheitsprobleme,
- Datenverlust,
- Unterbrechung der Produktion,
- Beeinträchtigung mehrerer Standorte.

Deshalb sollte vor der Umsetzung geprüft werden,

ob der erwartete Nutzen die möglichen Risiken rechtfertigt.

Was ist eine Change Authority?

Die **Change Authority** ist die Person oder Gruppe,

die über einen Change entscheidet.

ITIL schreibt **keine feste Rolle** vor.

Je nach Organisation kann dies beispielsweise sein:

- Teamleiter,
- Service Owner,
- Change Manager,
- Fachbereich,
- Projektleitung,
- IT-Leitung.

Die Entscheidung richtet sich nach Risiko und Auswirkungen des Changes.

Nicht jeder Change benötigt dieselbe Genehmigung

Ein Standard Change ist bereits vorab genehmigt.

Ein Normal Change benötigt häufig eine individuelle Freigabe.

Ein Emergency Change nutzt meist ein beschleunigtes Genehmigungsverfahren.

Die Genehmigung sollte immer zum Risiko passen.

Genehmigungen nach Risiko

Ein mögliches Beispiel:

Risiko	Mögliche Change Authority
gering	Teamleitung
mittel	Change Manager
hoch	Change Authority oder CAB
sehr hoch	Management oder Geschäftsleitung

Die konkrete Zuordnung legt jede Organisation selbst fest.

Was ist ein Change Advisory Board (CAB)?

Ein **Change Advisory Board (CAB)** ist ein beratendes Gremium.

Es unterstützt die Change Authority,

indem es Änderungen bewertet und Empfehlungen ausspricht.

Das CAB entscheidet **nicht automatisch selbst** über jeden Change.

Die endgültige Entscheidung trifft die zuständige Change Authority.

Aufgaben des CAB

Das CAB unterstützt beispielsweise bei:

- Risikobewertung,
- Auswirkungsanalyse,
- Terminplanung,
- Ressourcenplanung,
- Konflikterkennung,
- Priorisierung,
- Abstimmung zwischen Teams.

Dadurch verbessert sich die Qualität von Entscheidungen.

Mögliche Teilnehmer eines CAB

Je nach Change können unterschiedliche Personen beteiligt sein.

Beispiele:

- Change Manager,
- Service Owner,
- Systemadministratoren,
- Netzwerkadministratoren,
- Information Security,
- Fachbereiche,
- Projektleitung,
- Lieferanten.

Nicht jeder Teilnehmer ist bei jedem Change erforderlich.

Emergency CAB (ECAB)

Für besonders dringende Änderungen kann ein **Emergency Change Advisory Board (ECAB)** eingesetzt werden.

Es besteht meist aus wenigen entscheidungsbefugten Personen.

Dadurch können dringende Entscheidungen schneller getroffen werden.

Typische Situationen:

- kritische Sicherheitslücke,
- Produktionsausfall,
- Ransomware-Angriff,
- schwerwiegender Netzwerkausfall.

Der Change-Kalender

Ein Change-Kalender dokumentiert,

wann geplante Änderungen stattfinden.

Typische Informationen:

- Termin,
- betroffene Services,
- Verantwortliche,
- Wartungsfenster,
- Status,
- Risiko.

Dadurch lassen sich Überschneidungen vermeiden.

Vorteile eines Change-Kalenders

Ein Change-Kalender hilft dabei,

- Kollisionen zu vermeiden,
- Wartungsfenster zu koordinieren,
- Ressourcen besser zu planen,
- betroffene Teams rechtzeitig zu informieren,
- Risiken zu reduzieren.

Er verbessert die Übersicht über geplante Änderungen.

Beispiel eines Change-Kalenders

Datum	Service	Art des Changes
12.09.	VPN	Softwareupdate
13.09.	ERP	Datenbankupdate
14.09.	Firewall	Firmwareupdate
15.09.	WLAN	Controller-Aktualisierung

So erkennen Administratoren frühzeitig mögliche Konflikte.

Warum Terminüberschneidungen problematisch sind

Werden mehrere kritische Änderungen gleichzeitig durchgeführt,

kann dies zu Problemen führen.

Beispiel:

- Firewall-Update,
- Core-Switch-Austausch,
- Storage-Migration

am selben Abend.

Bei einer Störung wird die Ursachenanalyse deutlich erschwert.

Der Change-Kalender hilft,

solche Situationen zu vermeiden.

Wartungsfenster

Viele Changes werden innerhalb geplanter Wartungsfenster durchgeführt.

Typische Vorteile:

- geringere Auswirkungen auf Benutzer,
- besser planbare Arbeiten,
- ausreichende Zeit für Rollback,
- koordinierte Kommunikation.

Wartungsfenster sollten frühzeitig angekündigt werden.

Kommunikation vor einem Change

Vor größeren Änderungen sollten relevante Personen informiert werden.

Beispiele:

- Benutzer,
- Service Desk,
- Administratoren,
- Management,
- externe Dienstleister.

Die Information sollte unter anderem enthalten:

- Zeitpunkt,
- betroffene Services,
- erwartete Auswirkungen,

- Ansprechpartner.

Dokumentation der Genehmigung

Jede Genehmigung sollte nachvollziehbar dokumentiert werden.

Beispiele:

- wer genehmigt hat,
- Zeitpunkt,
- Risiko,
- Bedingungen,
- besondere Auflagen.

Dadurch bleiben Entscheidungen transparent.

Praxisbeispiel

Ein Unternehmen plant,

die Firmware aller Core-Switches zu aktualisieren.

Die Risikobewertung ergibt:

- hoher Business Impact,
- mehrere Standorte betroffen.

Der Change wird:

- im CAB besprochen,
- in den Change-Kalender eingetragen,
- für ein Wartungsfenster am Wochenende geplant.

Alle betroffenen Fachbereiche werden vorab informiert.

Typische Fehler

Fehler 1

Änderungen werden ohne Genehmigung durchgeführt.

Fehler 2

Der Change-Kalender wird nicht gepflegt.

Fehler 3

Mehrere kritische Changes finden gleichzeitig statt.

Fehler 4

Das falsche Entscheidungsgremium wird beteiligt.

Fehler 5

Emergency Changes werden nicht dokumentiert.

Fehler 6

Benutzer werden nicht informiert.

Fehler 7

Genehmigungen sind nicht nachvollziehbar.

Fehler 8

Wartungsfenster werden nicht eingehalten.

Fehler 9

Externe Dienstleister werden zu spät eingebunden.

Fehler 10

CAB-Sitzungen beschäftigen sich mit Standard Changes,

die bereits vorab genehmigt sind.

Checkliste Genehmigung

- [ ] Change-Typ bestimmt
- [ ] Risiko bewertet
- [ ] Change Authority festgelegt
- [ ] Genehmigung dokumentiert

- [ ] Verantwortlichkeiten bekannt
- [ ] Bedingungen berücksichtigt
- [ ] Kommunikation geplant
- [ ] Umsetzung freigegeben

Checkliste Change-Kalender

- [ ] Termin eingetragen
- [ ] Wartungsfenster abgestimmt
- [ ] Überschneidungen geprüft
- [ ] betroffene Services dokumentiert
- [ ] Verantwortliche bekannt
- [ ] Benutzer informiert
- [ ] Rollback berücksichtigt
- [ ] Monitoring vorbereitet

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker sind häufig an der Vorbereitung und Umsetzung von Changes beteiligt.

Typische Aufgaben:

- Risiken einschätzen,
- technische Informationen bereitstellen,
- Wartungsfenster abstimmen,
- Change-Kalender pflegen,
- Änderungen dokumentieren,
- Benutzer informieren,
- Monitoring nach dem Change überwachen.

Sie liefern wichtige Entscheidungsgrundlagen für die Change Authority.

Zusammenfassung

> Change planen
>
> ↓

> Risiko bewerten
>
> ↓
>
> Change Authority bestimmen
>
> ↓
>
> Genehmigung einholen
>
> ↓
>
> Change-Kalender aktualisieren
>
> ↓
>
> Beteiligte informieren
>
> ↓
>
> Umsetzung vorbereiten

Merksätze

> Die Change Authority entscheidet über einen Change.

> Das CAB unterstützt die Entscheidung – es ersetzt sie nicht.

> Standard Changes benötigen normalerweise keine erneute Genehmigung.

> Ein gepflegter Change-Kalender verhindert Konflikte zwischen Änderungen.

> Gute Kommunikation reduziert Risiken und Überraschungen.

Verwandte Seiten

- 13.1 Ziele und Grundlagen des Change Enablement
- 13.2 Change-Typen und Risikobewertung
- 13.4 Umsetzung, Tests und Rollback
- 13.5 Zusammenspiel mit Incident, Problem, Service Configuration Management und Monitoring
- Service Configuration Management
- Incident Management
- Monitoring and Event Management

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Change Enablement
- ITIL Foundation – Version 5

Einordnung

ITIL verwendet den Begriff **Change Authority** für die Person oder Gruppe, die einen Change genehmigt. Ein **Change Advisory Board (CAB)** besitzt in ITIL eine beratende Funktion und unterstützt die Entscheidungsfindung. Der Einsatz eines CAB, eines Emergency CAB (ECAB) und eines Change-Kalenders richtet sich nach Größe, Komplexität und Risiko der jeweiligen Organisation.

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

13.4 Umsetzung, Tests und Rollback

> **Kurz erklärt**
>
> Nach der Genehmigung beginnt die eigentliche Umsetzung eines Changes.
>
> Ziel ist es, Änderungen kontrolliert, nachvollziehbar und mit möglichst geringem Risiko einzuführen.
>
> Dazu gehören insbesondere sorgfältige Tests, eine geplante Durchführung, eine Überprüfung des Ergebnisses sowie ein vorbereiteter Rollback für den Fall unerwarteter Probleme.

Warum Tests wichtig sind

Auch sorgfältig geplante Änderungen können unerwartete Auswirkungen haben.

Beispiele:

- Anwendungen starten nicht mehr,
- Benutzer können sich nicht anmelden,
- Netzwerkverbindungen funktionieren nicht,
- Zertifikate werden nicht akzeptiert,
- Datenbanken reagieren langsamer.

Tests helfen, solche Probleme möglichst früh zu erkennen.

Testumgebungen

Änderungen sollten nach Möglichkeit zuerst in einer Testumgebung geprüft werden.

Typische Umgebungen:

- Entwicklungsumgebung (Development)
- Testumgebung (Test)
- Integrationsumgebung
- Staging
- Produktivsystem

Je ähnlicher die Testumgebung der Produktivumgebung ist,

desto aussagekräftiger sind die Testergebnisse.

Warum nicht direkt in der Produktion testen?

Tests im Produktivsystem können zu:

- Serviceunterbrechungen,
- Datenverlust,
- Sicherheitsproblemen,
- unerwarteten Ausfällen

führen.

Deshalb sollten Änderungen möglichst vorab getestet werden.

Arten von Tests

Je nach Change können unterschiedliche Tests erforderlich sein.

Beispiele:

- Funktionstest,
- Integrationstest,
- Lasttest,
- Sicherheitstest,
- Regressionstest,
- Benutzertest,
- Wiederherstellungstest.

Nicht jeder Change benötigt alle Testarten.

Funktionstest

Beim Funktionstest wird geprüft,

ob die neue oder geänderte Funktion wie vorgesehen arbeitet.

Beispiel:

Nach einer VPN-Aktualisierung wird getestet,

ob sich Benutzer erfolgreich verbinden können.

Regressionstest

Ein Regressionstest überprüft,

ob bestehende Funktionen nach einer Änderung weiterhin ordnungsgemäß arbeiten.

Beispiel:

Nach einem Datenbankupdate funktionieren weiterhin:

- Anmeldung,
- Berichtswesen,
- Druckfunktionen,
- API-Schnittstellen.

Regressionstests verhindern unbeabsichtigte Nebenwirkungen.

Pilotbetrieb

Bei größeren Änderungen erfolgt häufig zunächst ein Pilotbetrieb.

Dabei wird die Änderung nur für einen kleinen Benutzerkreis eingeführt.

Vorteile:

- geringeres Risiko,
- frühes Feedback,
- Fehler lassen sich leichter erkennen,
- Auswirkungen bleiben begrenzt.

Nach erfolgreichem Pilotbetrieb erfolgt die Einführung für alle Benutzer.

Wartungsfenster

Produktive Änderungen werden häufig innerhalb geplanter Wartungsfenster umgesetzt.

Vorteile:

- geringere Auswirkungen,
- ausreichende Zeit für Tests,
- genügend Zeit für Rollback,
- bessere Planbarkeit.

Größere Änderungen erfolgen häufig außerhalb der Hauptarbeitszeiten.

Durchführung des Changes

Während der Umsetzung sollten alle Schritte dokumentiert werden.

Typische Inhalte:

- Beginn,
- durchgeführte Arbeiten,
- Besonderheiten,
- Testergebnisse,
- Abweichungen,
- Abschlusszeit.

Eine vollständige Dokumentation erleichtert spätere Analysen.

Monitoring nach der Umsetzung

Nach Abschluss überwacht das Monitoring,

ob der Service ordnungsgemäß arbeitet.

Beispiele:

- Erreichbarkeit,
- CPU-Auslastung,
- Speicherverbrauch,
- Fehlermeldungen,
- Antwortzeiten,
- Netzwerkstatus.

Monitoring bestätigt den Erfolg des Changes.

Was ist ein Rollback?

Ein Rollback stellt den vorherigen Zustand wieder her,

wenn der Change nicht erfolgreich war.

Das Ziel besteht darin,

die Auswirkungen eines fehlgeschlagenen Changes möglichst gering zu halten.

Wann wird ein Rollback durchgeführt?

Beispiele:

- kritische Anwendungen funktionieren nicht,
- Service ist nicht verfügbar,
- Sicherheitsprobleme entstehen,

- Performance verschlechtert sich erheblich,
- unerwartete Fehler treten auf.

Nicht jeder Fehler führt sofort zu einem Rollback.

Die Entscheidung richtet sich nach Risiko und Auswirkungen.

Rollback-Plan

Ein Rollback sollte bereits vor Beginn des Changes vorbereitet werden.

Typische Inhalte:

- Auslösekriterien,
- Verantwortliche,
- Reihenfolge der Schritte,
- Wiederherstellung aus Backup,
- Zeitbedarf,
- Kommunikation.

Ein Rollback darf nicht erst während einer Störung geplant werden.

Beispiel eines Rollbacks

Ein klar definierter Ablauf verkürzt die Wiederherstellungszeit.

Backups vor einem Change

Vor vielen Änderungen wird ein Backup erstellt.

Beispiele:

- virtuelle Maschine,
- Datenbank,
- Konfiguration,
- Firewall,
- Netzwerkgerät,
- Active Directory.

Backups erleichtern den Rollback erheblich.

Verifizierung

Nach erfolgreicher Umsetzung wird geprüft,

ob der Change tatsächlich erfolgreich war.

Typische Fragen:

- Funktioniert der Service?
- Sind alle Tests erfolgreich?
- Gibt es Fehlermeldungen?
- Ist das Monitoring unauffällig?
- Wurde die Dokumentation aktualisiert?

Erst danach gilt der Change als abgeschlossen.

Post Implementation Review (PIR)

Nach größeren Änderungen wird häufig ein **Post Implementation Review (PIR)** durchgeführt.

Dabei wird bewertet:

- Wurde das Ziel erreicht?
- Gab es Probleme?
- War die Planung ausreichend?
- Hat der Rollback funktioniert?
- Welche Verbesserungen sind möglich?

Der PIR dient der kontinuierlichen Verbesserung zukünftiger Changes.

Dokumentation aktualisieren

Nach erfolgreichem Change sollten unter anderem aktualisiert werden:

- CMDB,
- Netzpläne,
- Runbooks,
- Architekturdiagramme,
- Knowledge Base,
- Betriebshandbücher.

Dadurch bleiben Informationen aktuell.

Praxisbeispiel

Ein Unternehmen aktualisiert seine zentrale Firewall.

Vorher:

- Backup erstellt,
- Rollback vorbereitet,
- Testplan erstellt.

Nach der Aktualisierung:

- Funktionstest erfolgreich,
- Monitoring ohne Auffälligkeiten,
- VPN-Verbindungen geprüft,
- Dokumentation aktualisiert.

Der Change wird anschließend erfolgreich abgeschlossen.

Typische Fehler

Fehler 1

Änderungen werden direkt produktiv getestet.

Fehler 2

Es existiert kein Rollback.

Fehler 3

Backups fehlen.

Fehler 4

Monitoring überprüft den Change nicht.

Fehler 5

Regressionstests werden ausgelassen.

Fehler 6

Dokumentationen werden nicht aktualisiert.

Fehler 7

Benutzer werden nicht informiert.

Fehler 8

Der Change wird zu früh abgeschlossen.

Fehler 9

Erfahrungen werden nicht ausgewertet.

Fehler 10

Testumgebung unterscheidet sich stark von der Produktivumgebung.

Checkliste Umsetzung

- [ ] Genehmigung vorhanden
- [ ] Testumgebung vorbereitet
- [ ] Wartungsfenster aktiv
- [ ] Monitoring eingerichtet
- [ ] Verantwortlichkeiten bekannt
- [ ] Dokumentation vorbereitet
- [ ] Kommunikation erfolgt
- [ ] Durchführung protokolliert

Checkliste Rollback

- [ ] Rollback definiert
- [ ] Backup vorhanden
- [ ] Verantwortliche benannt
- [ ] Zeitbedarf bekannt
- [ ] Wiederherstellung getestet
- [ ] Monitoring berücksichtigt
- [ ] Dokumentation vorhanden
- [ ] Auslösekriterien definiert

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker setzen die meisten Changes praktisch um.

Typische Aufgaben:

- Testsysteme vorbereiten,
- Backups erstellen,
- Änderungen durchführen,
- Monitoring überwachen,
- Funktionstests durchführen,
- Rollback vorbereiten,
- Dokumentationen aktualisieren,
- Erfahrungen für zukünftige Changes festhalten.

Eine sorgfältige Umsetzung entscheidet häufig über den Erfolg eines Changes.

Zusammenfassung

> Genehmigung erhalten
>
> ↓
>
> Tests durchführen
>
> ↓
>
> Backup erstellen
>
> ↓
>
> Change umsetzen

↓

Monitoring überprüfen

↓

Bei Bedarf Rollback durchführen

↓

Dokumentation aktualisieren

↓

Post Implementation Review durchführen

Merksätze

> Änderungen sollten möglichst nicht direkt in der Produktivumgebung getestet werden.

> Ein Rollback muss vor Beginn des Changes geplant sein.

> Monitoring bestätigt den Erfolg einer Änderung.

> Dokumentationen gehören zum Abschluss jedes Changes.

> Erfahrungen aus abgeschlossenen Changes verbessern zukünftige Änderungen.

Verwandte Seiten

- 13.1 Ziele und Grundlagen des Change Enablement
- 13.2 Change-Typen und Risikobewertung

- 13.3 Genehmigungen, Change Authority, CAB und Change-Kalender
- 13.5 Zusammenspiel mit Incident, Problem, Service Configuration Management und Monitoring
- Service Configuration Management
- Monitoring and Event Management
- Continual Improvement

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Change Enablement
- ITIL Foundation – Version 5

Einordnung

Die beschriebenen Vorgehensweisen zu Tests, Rollback, Monitoring und Post Implementation Review orientieren sich an den Empfehlungen der ITIL Practice „Change Enablement“. ITIL schreibt keine festen Testverfahren vor, empfiehlt jedoch ausdrücklich eine risikoorientierte Planung, geeignete Testmaßnahmen und eine nachvollziehbare Nachbereitung von Änderungen.

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

13.5 Zusammenspiel mit Incident, Problem, Service Configuration Management und Monitoring

> **Kurz erklärt**
>
> Change Enablement arbeitet eng mit vielen anderen ITIL Practices zusammen.
>
> Änderungen entstehen häufig aus Incidents oder Problems, beeinflussen Configuration Items und werden durch Monitoring überwacht.
>
> Erst das Zusammenspiel dieser Practices ermöglicht sichere Änderungen, stabile IT-Services und eine kontinuierliche Verbesserung der gesamten IT-Landschaft.

Warum das Zusammenspiel wichtig ist

Ein Change wird selten isoliert durchgeführt.

Vor einer Änderung müssen beispielsweise folgende Fragen beantwortet werden:

- Warum ist die Änderung notwendig?
- Welche Services sind betroffen?
- Welche Configuration Items werden verändert?
- Welche Risiken bestehen?
- Wie wird der Erfolg überprüft?

Diese Informationen stammen aus verschiedenen ITIL Practices.

Zusammenspiel der Practices

```
    |
    ▼
Service Configuration Management
    |
    ▼
Monitoring
    |
    ▼
Continual Improvement
```

Jede Practice liefert wichtige Informationen für die nächste.

Change Enablement und Incident Management

Viele Changes entstehen als Folge eines Incidents.

Beispiel:

Ein Server fällt regelmäßig aus.

Incident Management stellt den Betrieb zunächst wieder her.

Die eigentliche Ursache bleibt jedoch bestehen.

Zur dauerhaften Lösung wird ein Change geplant.

Incident Management arbeitet also häufig kurzfristig,

während Change Enablement eine nachhaltige Verbesserung ermöglicht.

Beispiel

```
Serverausfall
    |
    ▼
Incident Management
    |
    ▼
Service wiederhergestellt
    |
    ▼
```

Der Incident endet mit der Wiederherstellung,

der Change beseitigt anschließend die eigentliche Ursache.

Change Enablement und Problem Management

Problem Management untersucht,

warum Incidents wiederholt auftreten.

Wird eine dauerhafte Lösung gefunden,

führt diese häufig zu einem Change.

Beispiele:

- Software aktualisieren,
- Hardware austauschen,
- Netzwerk erweitern,
- Architektur ändern.

Problem Management liefert also häufig den Anlass für einen Change.

Known Errors

Ein Known Error kann eine empfohlene Änderung enthalten.

Beispiel:

Bekannter Fehler:

Firmware-Version 5.2 verursacht Abstürze.

Empfohlene Lösung:

Firmware-Version 5.3 installieren.

Die eigentliche Umsetzung erfolgt anschließend über Change Enablement.

Change Enablement und Service Configuration Management

Vor jeder Änderung muss bekannt sein,

welche Configuration Items betroffen sind.

Die CMDB liefert beispielsweise:

- betroffene Server,
- Anwendungen,
- Datenbanken,
- Netzwerke,
- Verantwortliche,
- Beziehungen.

Dadurch können Auswirkungen besser bewertet werden.

Impact Analysis

Vor einem Change wird häufig geprüft,

welche Services betroffen sind.

Beispiel:

```
Firewall
    |
    ├── VPN
    ├── Internet
    ├── VoIP
    └── E-Mail
```

Ein Firmware-Update betrifft somit mehrere Services.

Diese Information stammt aus der CMDB.

CMDB aktualisieren

Nach erfolgreicher Umsetzung werden unter anderem aktualisiert:

- Versionen,
- Status,
- Beziehungen,
- Dokumentationen,

- Verantwortlichkeiten.

Dadurch bleibt die Configuration-Datenbasis aktuell.

Change Enablement und Monitoring

Monitoring überwacht,

ob Änderungen erfolgreich umgesetzt wurden.

Beispiele:

- Server erreichbar,
- CPU normal,
- Speicher ausreichend,
- Netzwerk stabil,
- keine neuen Alarme,
- Services verfügbar.

Monitoring bestätigt somit den Erfolg eines Changes.

Monitoring vor einem Change

Auch vor einer Änderung liefert Monitoring wichtige Informationen.

Beispiele:

- aktuelle Auslastung,
- bekannte Fehler,
- bestehende Alarme,
- Performance,
- Verfügbarkeit.

Diese Informationen unterstützen die Risikobewertung.

Monitoring nach einem Change

Nach der Umsetzung werden häufig überprüft:

- Erreichbarkeit,
- Antwortzeiten,
- Fehlermeldungen,
- Auslastung,
- Benutzeranmeldungen,
- Dienste.

Erst wenn diese Prüfungen erfolgreich sind,

gilt der Change als abgeschlossen.

Change Enablement und Release Management

Ein Release kann mehrere Changes enthalten.

Beispiel:

Ein Software-Release umfasst:

- neue Funktionen,
- Fehlerbehebungen,
- Sicherheitsupdates,
- Konfigurationsänderungen.

Change Enablement bewertet und genehmigt die Änderungen,

Release Management plant und verteilt das Gesamtpaket.

Change Enablement und Deployment Management

Nach der Genehmigung erfolgt häufig die technische Bereitstellung.

Deployment Management übernimmt beispielsweise:

- Software verteilen,
- Container aktualisieren,
- Images bereitstellen,
- Anwendungen installieren.

Change Enablement steuert den organisatorischen Rahmen,

Deployment Management die technische Umsetzung.

Change Enablement und Information Security Management

Viele Changes dienen der Verbesserung der Sicherheit.

Beispiele:

- Sicherheitsupdates,
- Firewallregeln,
- MFA einführen,

- Zertifikate erneuern,
- Verschlüsselung aktivieren.

Dabei müssen Sicherheitsanforderungen und Serviceverfügbarkeit gleichermaßen berücksichtigt werden.

Change Enablement und Continual Improvement

Erfahrungen aus abgeschlossenen Changes fließen in Continual Improvement ein.

Beispiele:

- Planung verbessern,
- Testverfahren erweitern,
- Dokumentation optimieren,
- Genehmigungen vereinfachen,
- Standard Changes definieren.

Dadurch entwickelt sich der Change-Prozess kontinuierlich weiter.

Praxisbeispiel

Ein Unternehmen stellt fest,

dass nach jedem Betriebssystemupdate ähnliche Probleme auftreten.

Problem Management analysiert die Ursache.

Es zeigt sich,

dass wichtige Regressionstests fehlen.

Change Enablement ergänzt daraufhin den Testplan.

Monitoring bestätigt,

dass zukünftige Updates deutlich störungsärmer verlaufen.

Die neue Vorgehensweise wird anschließend als Standardprozess dokumentiert.

Typische Fehler

Fehler 1

Changes werden unabhängig von Incidents geplant.

Fehler 2

Configuration Items werden nicht aktualisiert.

Fehler 3

Monitoring überprüft Änderungen nicht.

Fehler 4

Problem Management wird nicht einbezogen.

Fehler 5

Release und Change werden verwechselt.

Fehler 6

Deployment erfolgt ohne Genehmigung.

Fehler 7

Auswirkungen auf andere Services werden unterschätzt.

Fehler 8

Erfahrungen aus früheren Changes werden nicht genutzt.

Fehler 9

CMDB und Monitoring liefern widersprüchliche Informationen.

Fehler 10

Verbesserungsmaßnahmen werden nicht dokumentiert.

Checkliste Zusammenspiel

- [ ] Incident analysiert
- [ ] Problem bewertet

- [ ] Change genehmigt
- [ ] Configuration Items geprüft
- [ ] Monitoring vorbereitet
- [ ] Dokumentation aktualisiert
- [ ] Erfolg überprüft
- [ ] Verbesserungen dokumentiert

Checkliste erfolgreicher Change

- [ ] Risiko bewertet
- [ ] Tests erfolgreich
- [ ] Rollback vorbereitet
- [ ] Monitoring aktiv
- [ ] CMDB aktualisiert
- [ ] Benutzer informiert
- [ ] Post Implementation Review durchgeführt
- [ ] Lessons Learned dokumentiert

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker begleiten nahezu den gesamten Lebenszyklus eines Changes.

Typische Aufgaben:

- Incidents analysieren,
- Ursachen dokumentieren,
- Configuration-Daten pflegen,
- Monitoring überwachen,
- Änderungen durchführen,
- Rollback vorbereiten,
- Dokumentationen aktualisieren,
- Verbesserungen ableiten.

Dadurch leisten sie einen wesentlichen Beitrag zu einem sicheren und stabilen IT-Betrieb.

Zusammenfassung

> “ Incident erkennen

↓

Ursache analysieren

↓

Change planen

↓

Auswirkungen bewerten

↓

Änderung umsetzen

↓

Monitoring prüfen

↓

CMDB aktualisieren

↓

Verbesserungen übernehmen

Merksätze

> Viele Changes entstehen aus Incidents oder Problems.

> Die CMDB liefert die Grundlage für Impact-Analysen.

> Monitoring bestätigt den Erfolg eines Changes.

>

> Deployment setzt Änderungen technisch um – Change Enablement steuert sie organisatorisch.

> Erfahrungen aus abgeschlossenen Changes verbessern zukünftige Änderungen.

Verwandte Seiten

- 13.1 Ziele und Grundlagen des Change Enablement
- 13.2 Change-Typen und Risikobewertung
- 13.3 Genehmigungen, Change Authority, CAB und Change-Kalender
- 13.4 Umsetzung, Tests und Rollback
- Incident Management
- Problem Management
- Service Configuration Management
- Monitoring and Event Management
- Release Management
- Deployment Management
- Continual Improvement

Quellen und Versionsstand

Offizielle Grundlagen

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

Einordnung

Die dargestellten Zusammenhänge entsprechen den Empfehlungen der ITIL-Practices. Change Enablement koordiniert Änderungen organisatorisch und arbeitet dabei eng mit Incident Management, Problem Management, Service Configuration Management, Monitoring sowie weiteren Practices wie Release und Deployment Management zusammen.

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