

Change Illusion Journal

Project Management, Cutover Management & SAP Transformation

Vorabveröffentlichung

- Was ist die Change Illusion?
- Vorproduktive versus produktive Deliverables
- Outcome Delivery Structure – ein fehlender Begriff
- Lernen mit Dr. Tortuga



Deutsche Ausgabe



Anmerkung der Redaktion

Willkommen zur ersten Ausgabe des *Change Illusion Journal*. Die **Change Illusion** ist eine spezielle Ausprägung der umfassenderen **Scope Illusion**. Während die Scope Illusion den Irrglauben beschreibt, dass ein Scope bereits existiert, bevor die zugehörigen Liefergegenstände (*Deliverables*) definiert wurden, entsteht die Change Illusion, wenn Governance-Entscheidungen fälschlich als Product Scope interpretiert werden.

Die Scope Illusion bildet den übergeordneten Bezugsrahmen und zugleich das zentrale Thema dieser Ausgabe.

Seit mehr als 20 Jahren arbeite ich in SAP-Projekten, zunächst als ABAP-Entwickler, später als Projektleiter, Programmmanager und Cutover-Manager. In nahezu jedem Projekt trat dasselbe Muster auf: Termine und Budgets wurden verfehlt. Die Ursache liegt jedoch selten dort, wo klassische Studien zum Projektversagen suchen. In diesem Beitrag argumentiere ich, dass viele Probleme auf eine **Scope Illusion** zurückzuführen sind. Bevor wir zur eigentlichen Ursache schlecht funktionierender SAP-Projekte gelangen, müssen wir zunächst einige methodische Grundlagen und die zugrunde liegende Terminologie klären.

Inhalt

Anmerkung der Redaktion.....	2
Inhalt.....	2
Kapitel 1 Von der Outcome Delivery Structure (ODS) zu operativen (OD) und voroperativen Deliverables (POD).....	5
Kapitel 2 Requirements, Changes und die Change-Governance-Matrix (ChGM).....	14
Kapitel 3 Changes vs. Deliverables – Warum eine Sardinenfabrik offenbart, was im Projektmanagement oft übersehen wird	18
Kapitel 4 Die „Change Illusion“ in SAP-Projekten	24
Glossar.....	35

Impressum

The Change Illusion Journal – Ausgabe 1 (Herbst 2026) Halbjährlich erscheinende digitale Publikation

Herausgeber GESWO Editorial San Antonio, Encarnación, Paraguay | **Chefredakteur** Guido Nageldinger | **Kontakt** www.geswo.com publisher@geswo.com |

ISBN (wird noch ergänzt) **ISSN** (wird nach Zuweisung ergänzt)

Erscheinungsweise Halbjährlich (zweimal pro Jahr) | **Copyright** © 2026 GESWO Editorial. Alle Rechte vorbehalten. Diese Zeitschrift erscheint ausschließlich als digitale PDF-Ausgabe. Kein Teil dieser Publikation darf ohne schriftliche Genehmigung vervielfältigt oder verbreitet werden. | **Haftungsausschluss** Es wird keine Gewähr für die Richtigkeit, Vollständigkeit oder Aktualität der Inhalte übernommen. | **Markenzeichen** SAP, SAP Activate, Jira, ITIL, TOGAF, PMI und andere erwähnte Begriffe sind eingetragene Markenzeichen der jeweiligen Unternehmen. | **Produktion** Titelbild und Grafiken erstellt mit Canva. Übersetzungen unterstützt durch DeepL.

Die Scope Illusion – Warum SAP-Projekte scheitern

Von Guido Nageldinger

Anmerkung der Redaktion – Diese Ausgabe des „*Change Illusion Journal*“ vertieft die erstmals in der Blog-Reihe vorgestellten Ideen. Sie verfeinert das konzeptionelle Verständnis des Cutovers, indem sie ihn mit der Outcome Delivery Structure (ODS) und dem Product Scope verknüpft. Anhand neuer visueller Modelle – „*Cutover vs. formelle Übergabe*“ und „*Cutover-Deliverable-Flow*“ – etabliert das Journal eine systemische Perspektive auf Transformation und zeigt, dass der Cutover nicht nur ein technisches Ereignis ist, sondern ein soziotechnischer Übergang, der in den organisatorischen Wandel eingebettet ist. Die hier vorgestellte überarbeitete Definition markiert eine bedeutende Weiterentwicklung des Konzepts und positioniert den Cutover als strukturelles Phänomen, das Deliverables, Governance und operative Bereitschaft über Projekt-, Programm- und Portfolioebenen hinweg miteinander verbindet.

Zusammenfassung – Dieser Leitartikel untersucht, warum viele SAP- und Transformationsprojekte scheitern, obwohl sie etablierte Projektmanagement-Methoden befolgen. Er stellt die Konzepte der „Outcome Delivery Structure“ (ODS), der „Deliverables“, des „Cutover“ und der „Change Illusion“ als Rahmen vor, um zu verstehen, wie Requirements, Changes und Deliverables miteinander in Beziehung stehen. Die Titelgeschichte ist in vier Kapitel unterteilt. Das erste Kapitel führt eine Reihe neuer Begriffe ein, wie beispielsweise die Outcome Delivery Structure (ODS). „Cutover“ wird als der Übergang von voroperativen Deliverables (PODs, pre-operational deliverables) zu operativen Deliverables (Ods, operational deliverables) im Kontext der ODS definiert. Übergangs-Deliverables (TDs, transitional deliverables) werden als Wegbereiter dieses Transformationsprozesses vorgestellt. Das zweite Kapitel beleuchtet den Unterschied zwischen Requirements und Changes. Requirements definieren den Zielzustand („to be“). Changes hingegen definieren die Abweichung vom As-Is. Die Projektplanung konzentriert sich auf Changes, da diese die eigentliche Arbeit darstellen. Es wird argumentiert, dass Changes oft mit Requirements verwechselt werden. Die Change Governance Matrix (ChGM) und die Zuordnung von Requirements zu Changes werden eingeführt, um die n:n-Beziehung zwischen dem Zielzustand und den Anpassungen des As-Iss transparent und überschaubar zu machen. Dies ermöglicht es dem CAB, seine vorgesehene Rolle bei der Bewertung und Steuerung von Changes auf Lösungsebene zu erfüllen. Das dritte Kapitel untersucht die Beziehung zwischen den Deliverables genauer. Deliverables ergeben sich weder direkt aus Changes noch aus der Outcome Delivery Structure. Vielmehr entstehen sie aus dem Zusammenspiel beider. Changes definieren, was transformiert werden soll. Die ODS definiert, wie die Transformation strukturiert ist. Aus dieser Kombination entstehen die Deliverables. Die deduktive Gleichung verdeutlicht zudem, dass jedes Deliverable zu genau einer ODS gehört. Kapitel vier führt das Konzept der „Change Illusion“ ein, also die falsche Annahme, dass eine Entscheidung für eine Change automatisch den Scope definiert. Dies tritt auf, wenn Organisationen eine übergeordnete Governance-

Entscheidung („Wir brauchen eine neue Fabrik“, „Wir brauchen mehr Sicherheit“, „Wir brauchen Harmonisierung“) so behandeln, als hätte sie den tatsächlichen Scope bereits definiert. In Wirklichkeit definiert eine Change aber nicht den Product Scope. Es ist lediglich die Entscheidung, den aktuellen As-Is Zustand zu transformieren. Der Product Scope ergibt sich erst später durch die innerhalb der ausgewählten Outcome Delivery Structure (ODS) ausgearbeiteten Deliverables. Kapitel 4 untersucht zudem verschiedene sogenannte SAP-„Requirement-Workshops“ und kommt zu dem Schluss, dass es sich dabei tatsächlich um Change-workshops handelt. Die SAP-Activate-Methodik lässt die Ableitung von Deliverables außer Acht, da sie davon ausgeht, dass der Scope im Standardsystem bereits definiert ist und dass die Changes den Scope festlegen, woraufhin sich die Deliverables ganz natürlich ergeben. In Wirklichkeit ist dies falsch – und genau deshalb geraten SAP-Projekte aus der Bahn. Die Abhandlung kommt zu dem Schluss, dass viele Projekte scheitern, weil sie Changes statt Deliverables ausarbeiten (elaborieren). Es wird daher ein Vorprojekt empfohlen, um vor Beginn der Implementierung aus den ausgearbeiteten Changes die Deliverables abzuleiten und so den fehlenden Rahmen für Planung, Steuerung und Cutover-Management zu schaffen.

Kapitel 1

Von der Outcome Delivery Structure (ODS) zu operativen (OD) und voroperativen Deliverables (POD)



Abbildung1 : Vergessen Sie nicht, sich den ersten Teaser anzusehen, der auf der Zeitschriftenseite dieser Ausgabe aufgeführt ist!
<https://geswo.com/de/journal/issue01-2026/>

Jedes Unternehmen muss sich an Marktveränderungen anpassen und sein Geschäft transformieren. Wie? Durch Projekte, Programme und komplexe Transformationsinitiativen. Seit Jahrzehnten sprechen wir von Portfolio, Programm, Projekt und in jüngerer Zeit auch von „Epic“. Doch eines hat bisher gefehlt: ein gemeinsamer Oberbegriff, der diese Ebenen logisch zusammenfasst und beschreibt, wie Organisationen ihre Arbeit strukturieren, um ein Deliverable zu erzielen.

Genau hier kommt ein neuer Begriff ins Spiel:

Definition – Outcome Delivery Structure (ODS): Die Gesamtheit aller strukturellen Ebenen, die dazu dienen, ein Deliverable zu erzielen.

ODS ist ein moderner, Framework-neutraler Ansatz, der die bekannten Ebenen nicht ersetzt, sondern sie übersichtlicher ordnet.

ODS umfasst somit:

- klassische Projektstrukturen (PMI, PRINCE2)
- agile Strukturen (Scrum, SAFe)
- Hybridmodelle (SAP Activate)
- serviceorientierte Modelle (ITIL)

1.1. Klassische hierarchische Strukturen

Zu den klassischen hierarchischen Strukturen gehören:

- **Portfolio, Ebene 1** – strategische Sammlung aller Initiativen
- **Programm, Ebene 2** – koordinierte Gruppe verwandter Projekte
- **Epic, Ebene 3** – große Arbeitseinheit, die in Projekte unterteilt ist
- **Projekt, Ebene 4** – klar definiertes Deliverable mit Start- und Enddatum
- **Aufgabe, Ebene 5** – kleinste operative Einheit

1.2. Die agile Hierarchie (SAFe & Scrum)

In einer skalierten agilen Umgebung (**SAFe**) fügen sich die Begriffe nahtlos in die bestehende Struktur ein:

- **Epic (Portfolio-/Lösungs-/Programmebene):** steht ganz oben. Es beschreibt eine große, strategische Initiative (z. B. „Entwicklung eines neuen Kundenportals“).
- **Feature (Programmebene):** Ein Epic wird in Features unterteilt (z. B. „Anmeldesystem“ oder „Rechnungsübersicht“).
- **User Story (Teamebene):** Features werden zur Umsetzung in User Stories unterteilt (z. B. „Als Kunde möchte ich mich mit meiner E-Mail-Adresse anmelden können“).
- **Sprint (Zeitraumen):** Ein Sprint ist **keine hierarchische Arbeitseinheit**, sondern ein zeitbasierter Rahmen (in der Regel 2–4 Wochen). Innerhalb dieses Rahmens arbeitet das Team eine Auswahl von User Stories ab.

Visualisierung der SAFe-Hierarchie mit Zeiteinheiten:

1. **Portfolio / Lösung** (Strategie)
2. ↳ **Epic** (Monate bis Quartale)
3. ↳ **Feature** (passt in ein PI / Program Increment, ca. 8–12 Wochen)
4. ↳ **User Story** (passt in einen **Sprint**, einige Tage)

1.3. Klassifizierung in SAP Activate

SAP Activate verknüpft agile Konzepte mit traditionellen Projektphasen:

- **Epic:** Entspricht einem größeren **Deliverable** innerhalb eines *Arbeitsstroms* (z. B. im Arbeitsstrom „*Anwendungsdesign und -konfiguration*“).
- **User Story:** Im SAP-Kontext wird diese häufig während der *Explore-Phase* im **Backlog** erfasst. Sie verfeinert die Requirements und entspricht auf der untersten Ebene den **Aufgaben**, die im Sprint umgesetzt werden.
- **Sprint:** Findet vorwiegend in der „**Realize**“-Phase statt. Hier werden die im Backlog priorisierten User Stories in iterativen Zyklen (Sprints) umgesetzt und getestet.

1.4. Klassifizierung nach PMI und ITIL

- **PMI (Classic & Hybrid):** Ein **Epic** entspricht einer übergreifenden Phase oder einem großen Arbeitspaket im Projektstrukturplan (PSP/WBS). Die **User Story** ist die spezifische, detaillierte Aktivität. Der **Sprint** ersetzt die klassische, starre Zeitplanabfolge durch kurze, iterative Abschnitte.
- **ITIL (Wertströme):** Ein Epic und seine User Stories sind die spezifischen Requirements, die durch einen **Wertstrom** fließen. Die Umsetzung von User Stories innerhalb eines **Sprints** ist Teil der Aktivitäten „*Design & Transition*“ oder „*Obtain/Build*“ und liefert letztendlich einen Mehrwert für den Kunden.

1.5. Die konzeptionelle Lücke in *allen* Rahmenwerken : POD, OD und TD

Wir verfügen über perfekte Begriffe für strukturelle Ebenen (*Portfolio, Programm, Epic, Feature, Projekt, User Story, Aufgabe*), aber keinen einzigen guten Begriff für den Status des Liefergegenstands, solange dieser noch nicht im ODS in Produktion ist.

Ich habe in meinem Buch über SAP-Cutover-Management bereits klar definiert, was unter „*Deliverables*“ (Liefergegenständen) und „*Result Types*“ (Ergebnistypen) zu verstehen ist.

Definition: Deliverable

Ein Deliverable ist ein greifbares Dokument oder Artefakt (z. B. der vollständige Cutover-Plan), das zur Abnahme bereit, versioniert und formell genehmigt ist.

Anmerkung zur Terminologie:

Neben dem Begriff „*Deliverable*“ (*Liefergegenstand*) werden auch die Begriffe „**Result-types**“ (**Ergebnistyp**) und „**Outcome**“ (**Realisierter Zielzustand**) häufig verwendet. Diese Konzepte werden in diesem Zeitschriftenbeitrag nicht näher erläutert. Ein **Ergebnistyp** stellt in der Regel eine weniger formelle Komponente dar, die zu einem oder mehreren Deliverables beitragen kann, wie beispielsweise Sitzungsprotokolle, Workshop-Deliverables, Aktionspunkte oder Entscheidungsprotokolle. Ein **Deliverable** kann als der realisierte Zustand verstanden werden, der sich aus der Umsetzung einer oder mehrerer Changes durch eine Reihe von Deliverables ergibt.

Es gibt jedoch zwei Arten von Deliverables in einem Projekt. Einige beziehen sich auf den „**Project Scope**“, andere auf den „**Product Scope**“, um die Terminologie des PMI zu verwenden. Der **Projekt Scope** umfasst alle Deliverables, die zur Verwaltung, Steuerung und zum Abschluss des Projekts selbst erforderlich sind. Dazu gehören beispielsweise Projektpläne, Governance-Dokumente, Kommunikationsstrukturen, Statusberichte und Schulungsunterlagen. Der **Product Scope** umfasst die Deliverables, aus denen das eigentliche Produkt oder System besteht.

Nur die Deliverables, die sich auf den Product Scope beziehen, sind in der Regel in die Produktion übertragbar – das heißt, sie sind für den Cutover relevant. Man könnte hier pedantisch sein und argumentieren, dass einige Deliverables aus dem Projekt Scope tatsächlich übertragen werden könnten, wie beispielsweise gewonnene Erkenntnisse.

Beispiel: Der Systemintegrationstest (SIT) und der Benutzerakzeptanztest (UAT, User Acceptance Test) sind Deliverables innerhalb des Projekt Scopes. Auch wenn sie Akzeptanzkriterien für die Umstellung sind, werden sie nicht übertragen/transformiert. Systemkonfigurationen, neue Prozessabläufe, geschultes Personal und für den produktiven Einsatz bereite Dokumente sind übertragbare (Cutover-relevante) Deliverables.

Und hier ist der fehlende Begriff: Pre-Productive Deliverables (POD)

Definition: Vorbetriebliche Leistung (POD) im Vergleich zur betrieblichen Leistung (OD)

Ein in den produktiven Betrieb überführbarer Liefergegenstand (Deliverable) ist ein pre-productive Deliverable (POD), solange dieser noch nicht im Produktivbetrieb ist. Während des Cutovers wird ein POD in ein OD, also ein operativen und betriebsbereiten Liefergegenstand (Deliverable), überführt.

Interessanterweise erfordern nicht alle Projekte Pre-Productive Deliverables. Bei Bauprojekten wie Gebäuden oder Brücken ist das Produkt unmittelbar nach Fertigstellung betriebsbereit. Diese Projekte beinhalten daher keine technische Umstellung im IT-Sinne (Cutover). Stattdessen stützen sie sich auf formelle Übergabeaktivitäten wie die Schlüsselübergabe oder behördliche Genehmigungen.

Hybride Fälle lassen die Grenze verschwimmen. Im Schiffbau beispielsweise fungieren Probefahrten und Sicherheitsprüfungen als vorbetriebliche Meilensteine, während der feierliche Stapellauf und die Übergabe an die Reederei eine formelle Übergabe darstellen. In ähnlicher Weise erfordern Wasserkraftwerke Scopereiche vorbetriebliche Meilensteine (Testläufe, Netzsynchronisation, Sicherheitsvalidierung) und beinhalten zudem eine formelle Übergabe an den Betreiber. Dies wird in der folgenden Abbildung (Cutover-Handover-Matrix) und Tabelle veranschaulicht.

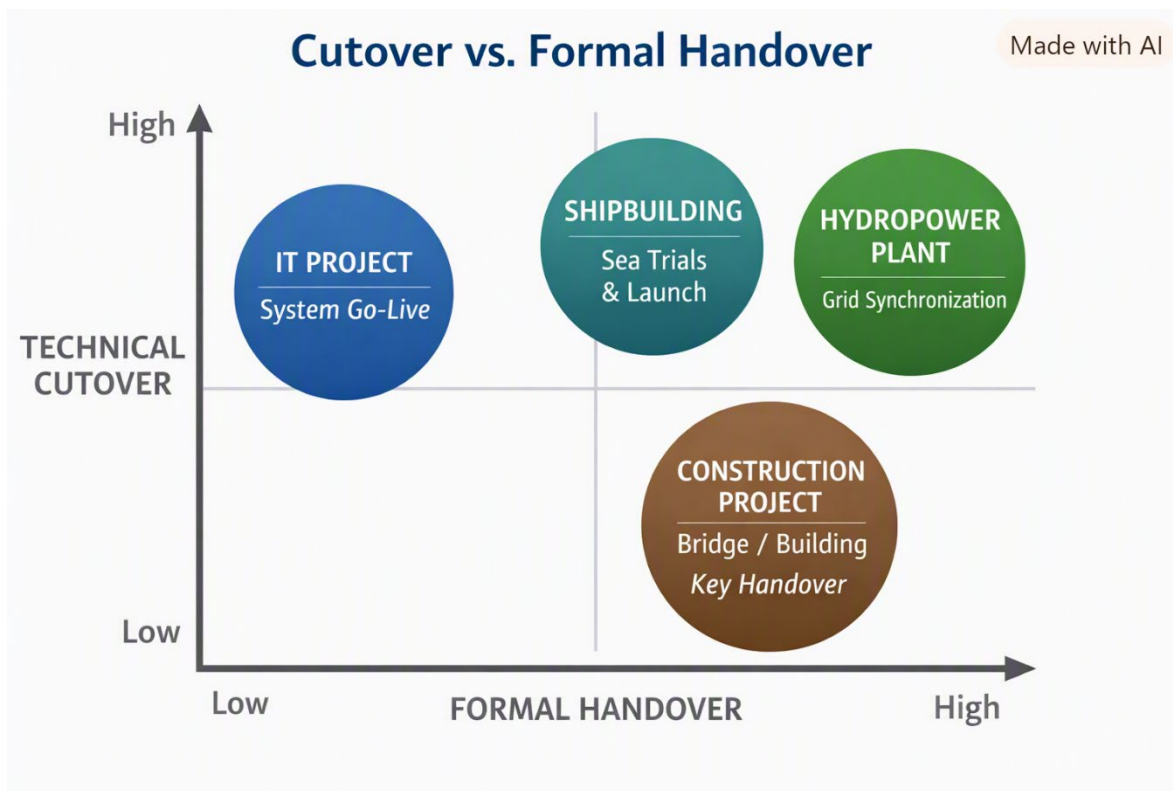


Abbildung2 : Cutover vs. formelle Übergabe Die Matrix veranschaulicht, wie verschiedene Projektarten den technischen Cutover und die formelle Übergabe gegeneinander abwägen. IT-Projekte legen den Schwerpunkt auf technische Übergänge, Bauprojekte konzentrieren sich auf die formelle Übergabe, während Schiffbau und Wasserkraftwerke beide Dimensionen kombinieren. Der technische Cutover umfasst sowohl die Systemaktivierung als auch die organisatorische Anpassung.

Tabelle1 : Vergleich – Cutover bei verschiedenen Projektarten

Projektart	Vorbetriebliche Leistungen (POD)	Art der Umstellung	Beispiel für die Übergabe	Anmerkungen
IT-Projekt	Ja, obligatorisch (z. B. Testpläne, Migrationsskripte, Datenvalidierung)	Technisch-organisatorische, klar definierte Umstellung	Go-Live, Systemumstellung	Klassisches Cutover-Szenario mit POD → OD → Deliverable
Bauprojekt (Brücke, Gebäude)	Nein, keine typischen PODs	Formelle Übergabe, nicht technischer Art	Schlüsselübergabe, behördliche Genehmigung	Das Produkt ist nach Fertigstellung betriebsbereit
Schiffbau	Teilweise: Docktests, Sicherheitsüberprüfungen, Schulung der Besatzung	Hybrid: technische und formale Umstellung	Probefahrten, Übergabe an die Reederei	Grenzen verschwimmen: Es gibt PODs, aber die Übergabe markiert auch die Fertigstellung des Produkts
Wasserkraftwerk	Ja, Scopereiche PODs (Testläufe, Netzanschluss, Sicherheitsprüfungen)	Technisch-organisatorische Umstellung mit hoher Komplexität	Netzsynchonisierung, Inbetriebnahme	Ähnelt stark einer IT-Umstellung: PODs sind entscheidend für den sicheren Betrieb

Der Begriff „**technische Umstellung**“ (**Technical Cutover**) bezieht sich auf den koordinierten Übergang von Systemen, Prozessen und Organisationsstrukturen von einem voroperativen in einen operativen Zustand. Während der Begriff die technische Umsetzung (Systemumstellung, Datenmigration, Konfiguration) betont, umfasst er von Natur aus auch organisatorische und verfahrenstechnische Changes, die erforderlich sind, um die neue Betriebsrealität aufrechtzuerhalten. Daher ist eine IT-Umstellung nicht rein technischer Natur – sie ist ein soziotechnisches Ereignis, das sowohl systemische als auch geschäftliche Changes mit sich bringt.

Die Cutover-Handover-Matrix führt eine neue Methode zur Klassifizierung von Projektübergängen ein. Sie visualisiert das Kontinuum zwischen technischer Inbetriebnahme und formeller Übergabe branchenübergreifend und zeigt, dass ein Cutover nicht rein technischer, sondern soziotechnischer Natur ist.

Fazit: Ein Cutover ist kein universelles Konzept, sondern ein Spektrum. IT-Projekte legen den Schwerpunkt auf den technischen Cutover, Bauprojekte auf die formelle Übergabe, und komplexe Ingenieurprojekte (Schiffe, Kraftwerke) kombinieren beide Dimensionen.

Es gibt noch eine weitere Art von Deliverable, die wir bisher noch nicht behandelt haben: Übergangs-Deliverables sind tatsächlich eigenständige Deliverables und spielen im Kontext des Cutover eine zentrale Rolle.

Definition: Übergangs-Deliverable (TD, Transitional Deliverables)

Übergangs-Deliverables sind temporäre Artefakte, die erstellt werden, um den Übergang vom voroperativen Zustand in die Betriebsumgebung zu ermöglichen. Sie sind nicht Teil des Endprodukts, dienen jedoch als wesentliche Hilfsmittel während der Cutover-Phase.

Beispiele hierfür sind:

- **IT-Kontext:** Übergangssysteme, temporäre Datenbanken, Migrationsskripte, Sandbox-Umgebungen.
- **Physische Projekte:** Bau eines Lagers, temporäre Infrastruktur, temporäre Lösungen für Test- oder Übergangsphasen.
- **Organisatorisch:** Schulungsplattformen, Übergangsprozesse, temporäre Rollen oder Governance-Strukturen.

Übergangsleistungen weisen folgende Merkmale auf:

- **Zweck:** Unterstützt die Umstellung, nicht das Endprodukt
- **Lebenszyklus:** Bestehen nur für die Dauer der Umstellung; werden außer Betrieb genommen, sobald die Betriebsstabilität erreicht ist.
- **Verantwortung:** Liegt häufig zwischen den Projekt- und Betriebsorganisationen
- **Wert:** Reduzierung von Risiken, Aufrechterhaltung der Servicekontinuität und Unterstützung der Bereitschaftsprüfung.

Cutover Deliverable Flow

Made with AI

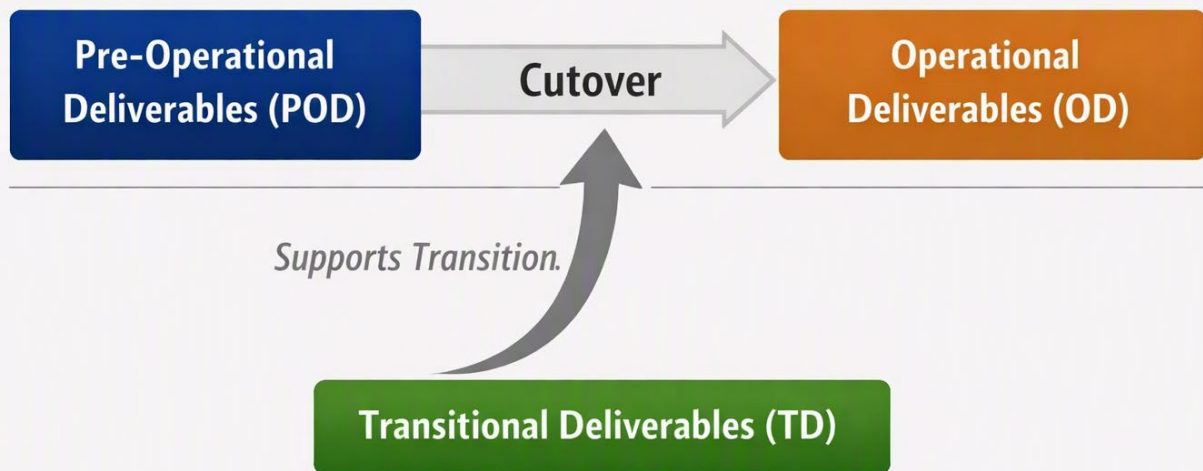


Abbildung3: Übergangs-Deliverables (TD) sind unterstützende Artefakte, die die Umstellung ermöglichen. Sie werden nicht in operative Deliverables (OD) umgewandelt, sondern stellen die vorübergehenden Strukturen, Werkzeuge oder Umgebungen bereit, die für die Umwandlung von voroperativen Deliverables (POD) in operative Deliverables (OD) erforderlich sind. In diesem Modell fließen die TDs in den Cutover-Prozess ein, anstatt Teil der Transformationskette zu sein. Ohne sie könnte der Cutover weder sicher noch effektiv erfolgen.

Während voroperative Deliverables die Umgebung vorbereiten und operative Deliverables das Endprodukt definieren, gewährleisten Übergangs-Deliverables einen kontrollierten und stabilen Übergang zwischen beiden. Sie verkörpern die temporäre Infrastruktur der Transformation – das Gerüst, das es ermöglicht, dass die neue operative Realität sicher entsteht, siehe Abbildung2 .

1.6. Übergangs-Deliverables und die Go-Live-Strategie

Übergangs-Deliverables (TDs) gehören zum Product Scope, auch wenn sie dort selten definiert werden. Diese Unterscheidung ist unerlässlich, um Verwechslungen zwischen Projektmanagement-Aktivitäten und Produkttransformations-Aktivitäten zu vermeiden. TDs sind Teil des Produktübergangs, nicht der Projektsteuerung, und entstehen während der Ausarbeitung der **Go-Live-Strategie**, die festlegt, wie voroperative Deliverables (PODs) in operative Deliverables (ODs) umgewandelt werden.

TDs sind nicht Teil des Produkts selbst, sondern wesentliche Voraussetzungen für den Cutover. Sie stellen temporäre Artefakte dar – wie Migrationsumgebungen, Fallback-Systeme oder provisorische Einrichtungen –, die den Übergang technisch und organisatorisch ermöglichen.

Da TDs erst bei der Festlegung der Go-Live-Strategie ausgearbeitet werden, spricht ihre Existenz dafür, diese Strategie vor Beginn der Realisierungsphase abzuschließen. Ohne eine klare Go-Live-Strategie

bleiben die TDs undefiniert, und das Projekt läuft Gefahr, ohne die erforderliche Übergangsinfrastruktur in die Realisierungsphase einzutreten.

Diese Unterscheidung hilft auch dabei, die „**Scope-Illusion**“ zu vermeiden: das Missverständnis, dass die Projektplanung abgeschlossen ist, sobald Aufgaben definiert sind, anstatt erst dann, wenn die Deliverables – einschließlich der Übergangs-Deliverables – vollständig verstanden sind. TDs zeigen, dass sich der Product Scope über das Endprodukt hinaus erstreckt und die Artefakte umfasst, die zur Erreichung der Betriebsbereitschaft erforderlich sind.

1.6. Zusammenfassung

Dieses Kapitel schließt mit einer Definition von „Cutover“ und fasst alle bisher eingeführten Begriffe zusammen.

Definition: Cutover

Der Cutover wandelt *voroperative Deliverables (Pre-Operational Deliverables, POD)* in *operative Deliverables (Operational Deliverables, OD)* um. Diese operative Umwandlung bezieht sich stets auf die ausgewählte *Outcome Delivery Structure (ODS)* und wird **durch Übergangs Deliverables (Transitional Deliverables, TD)** ermöglicht.

Die gewählte ODS erfordert nicht immer PODs und umfasst daher möglicherweise keinen Cutover, sondern endet stattdessen mit einer formellen operativen Übergabe – wie bei Bauprojekten.

PODs, ODs und TDs sind Teil des **Product Scopes**. TDs sind zu Beginn selten bekannt und werden während der Projektdurchführung ausgearbeitet, meist im Rahmen der *Go-Live-Strategie* vor der Realisierungsphase.

Die Definition hebt zudem hervor, dass ein Cutover nicht lediglich projektspezifisch, sondern ODS-abhängig ist. Er kann daher je nach der Struktur, auf die er sich bezieht, *projektspezifisch*, *programmspezifisch* oder *portfoliospezifisch* sein. Nicht jede ODS erfordert einen Cutover – aber jeder Cutover erfordert eine ODS.

Die Definition setzt Cutover-Aktivitäten mit der Transformation von PODs in Verbindung. Ein Cutover ist daher nicht nur eine Ansammlung technischer Systemaktivitäten. Es handelt sich um ein **soziotechnisches Ereignis**, das sowohl systemische als auch geschäftliche Changes zum Ausdruck bringt.

TDs zeigen, dass die Go-Live-Strategie kein spätes operatives Artefakt ist, sondern eine strategische Entwurfsaktivität, die vor Beginn der Realisierung abgeschlossen sein sollte. Ihr Auftauchen während der Go-Live-Planung offenbart die verborgene Ebene der Liefergegenstände im Product Scope, die die Transformation von PODs zu ODs ermöglichen – und deckt damit die „**Scope-Illusion**“ (ein Begriff, der im folgenden Kapitel näher erläutert wird) auf, wenn die Projektplanung sich nicht an den Liefergegenständen ausrichtet.

Kapitel 2

Requirements, Changes und die Change-Governance-Matrix (ChGM)



Abbildung4 : Vergessen Sie nicht, sich den zweiten Teaser anzusehen, der auf der Journalseite dieser Ausgabe aufgeführt ist!
<https://geswo.com/de/journal/issue01-2026/>

2.1. Requirements und Changes – Zwei Perspektiven auf denselben Zielzustand

In Projekten werden „Requirements“ und „Changes“ oft verwechselt, obwohl sie zwei unterschiedliche Perspektiven auf denselben Zielzustand darstellen. Requirements definieren den zukünftigen Zustand eines Systems oder Prozesses. Sie beschreiben, was in der Ziellösung vorhanden sein muss, unabhängig von der aktuellen Ist-Situation. Changes hingegen definieren die Abweichung vom As-Is – die konkrete Anpassung, die erforderlich ist, um den Zielzustand zu erreichen oder zu modifizieren. Diese Unterscheidung ist wesentlich, da sie die funktionale Definition des Zielzustands von den technischen oder organisatorischen Maßnahmen trennt, die zu dessen Erreichung erforderlich sind.

2.1.1. Requirements als Definition des Zielzustands (To-Be)

Eine Anforderung beschreibt, was ein System in der Ziellösung leisten muss. Sie entsteht typischerweise in frühen Projektphasen und ist lösungsneutral. Requirements definieren den Scope, die funktionalen Erwartungen und die Kriterien, die später im UAT akzeptiert werden. Sie sind nicht reaktiv, sondern planungsorientiert. Eine Anforderung wie „Das System muss vier Sprachen unterstützen“ beschreibt nur den Zielzustand, nicht die Arbeit, die erforderlich ist, um ihn zu erreichen.

2.1.2. Changes als Anpassungen am As-Is

Eine Change ergibt sich immer aus der Differenz zwischen dem As-Is und dem To-Be. Wenn ein System bereits Russisch und Spanisch unterstützt (As-Is), die Soll-Lösung jedoch vier Sprachen erfordert (To-Be), besteht die Change darin, Englisch und Französisch hinzuzufügen. Die Change ist somit die konkrete Anpassung des As-Is und nicht die abstrakte Definition des To-Be. Changes ergeben sich aus neuen Erkenntnissen, technischen Einschränkungen, regulatorischen Requirements oder Fehlern und haben direkte Auswirkungen auf Zeitplan, Budget, Tests und die Umstellung.

2.2. Warum UAT-Tests Requirements prüfen – und nicht Changes

Der User Acceptance Test (UAT) ist die Abnahme des Zielzustands (To-Be). Er testet nicht die Change selbst, sondern das Deliverable der Change. Selbst wenn das System bereits Russisch und Spanisch unterstützte, müssen diese Sprachen im UAT erneut getestet werden, da der UAT den vollständigen Zielzustand abnimmt. Der UAT testet daher nicht nur die neuen Sprachen Englisch und Französisch, sondern alle vier Sprachen, da die Anforderung vier Sprachen definiert. Der UAT ist kein Changetest, sondern ein Anforderungstest. Er bestätigt, dass der Zielzustand vollständig erreicht wurde – unabhängig davon, wie viele Changes erforderlich waren.

2.2.1. Regressionstests als integraler Bestandteil der UAT

Das erneute Testen bestehender Funktionen ist eine logische Konsequenz der Abnahme des Zielzustands. Jede Change kann sich auf bestehende Funktionen auswirken, sei es durch neue Konfigurationen, neue Datenmodelle, neue UI-Elemente oder neue Übersetzungslogik. Die UAT testet nicht die Historie; sie testet den Zielzustand – und dieser Zielzustand umfasst alle Funktionen, nicht nur die geänderten Teile.

2.3. Die Projektplanung konzentriert sich auf Changes – nicht auf Requirements

Requirements definieren den Zielzustand, erzeugen jedoch keinen Arbeitsaufwand. Erst die Abweichung vom As-Is, der Change, verursacht Aufwand, Ressourcenbedarf, Auswirkungen auf den Zeitplan und das Budget. Daher planen Projekte keine „Requirements“, sondern „Changes“. Die Projektplanung beantwortet die Frage: „Was müssen wir tun, um den Zielzustand zu erreichen?“ Und diese Frage lässt sich nur auf der Grundlage von Changes beantworten, nicht anhand von Requirements.

2.4. Warum das PMI hier nicht unterscheidet

In der klassischen PMI-Logik ist ein Changerequest eine Modifikation des genehmigten Projekt Scopes. Das PMI betrachtet Changes ausschließlich als Changes des Projekt Scopes und unterscheidet nicht zwischen Anforderungs Changes und Lösungs Changes. Für das PMI ist jeder Change eine Modifikation des Projekt Scopes, unabhängig davon, ob sie durch neue funktionale Requirements, technische Einschränkungen, regulatorische Vorgaben oder Mängel ausgelöst wird. Diese Sichtweise ist für das

traditionelle Projektmanagement ausreichend, spiegelt jedoch nicht die Komplexität moderner Lösungen wider.

2.5. Zuordnung von Requirements zu Changes – Die Verbindung zwischen To-Be und As-Is

In realen Projekten besteht kein linearer Zusammenhang zwischen Requirements und Changes. Es handelt sich stets um eine n:n-Beziehung: Eine Anforderung kann mehrere Changes nach sich ziehen, und eine Change kann sich auf mehrere Requirements auswirken. Eine Anforderung erfordert möglicherweise keine Change, wenn der As-Is bereits mit dem To-Be übereinstimmt. Eine Change kann ohne eine Anforderung entstehen, beispielsweise aufgrund gesetzlicher Vorgaben oder Fehler. Das Mapping von Requirements zu Changes macht diese Zusammenhänge sichtbar und bildet die Grundlage dafür, Changes nachvollziehbar, testbar und steuerbar zu machen.

Die Zuordnung zeigt, welche konkreten Changes zur Umsetzung einer Anforderung notwendig sind und welche Requirements von einem Change betroffen sind. Sie bildet die Grundlage für die Auswirkungsanalyse, die Testplanung und die Cutover-Bewertung. Ohne Zuordnung bleibt der Zusammenhang zwischen Sollzustand und Ist-Anpassungen unsichtbar – ebenso wie Risiken, Abhängigkeiten und Regressionen.

2.6. Warum das CAB oft Requirements statt Changes bewertet

In ITIL ist das Change Advisory Board (CAB) dafür vorgesehen, Changes zu bewerten – also die Abweichungen vom As-Is, die erforderlich sind, um den Zielzustand zu erreichen oder anzupassen. In der Praxis werden in CAB-Sitzungen jedoch oft Requirements statt Changes vorgestellt. Das CAB erörtert dann funktionale Zieldefinitionen, obwohl es eigentlich die Auswirkungen konkreter Changes bewerten sollte. Der Grund dafür ist einfach: Eine Zuordnung von Requirements zu Changes wird selten vorgenommen. Wenn unklar ist, welche Changes sich aus einer Anforderung ergeben, bleibt dem CAB nichts anderes übrig, als die Anforderung selbst zu bewerten.

Infolgedessen verliert das CAB seine eigentliche Funktion als Gremium zur Steuerung von Lösungs-Changes und wird zu einem Gremium zur Überprüfung des Scopes – ähnlich der klassischen PMI-Logik. Risiken, Abhängigkeiten, Regressionen und Auswirkungen der Umstellung bleiben unsichtbar, da die Changes nicht explizit dokumentiert wurden.

2.7. Die Change-Governance-Matrix (ChGM™) als Lösung

Die Change Governance Matrix (ChGM™) trennt diese Ebenen klar voneinander. Sie betrachtet Requirements als Definition des Zielzustands und Changes als Abweichungen vom As-Is. Erst durch die Zuordnung von Requirements zu Changes wird sichtbar, welche konkreten Changes zur Umsetzung einer Anforderung notwendig sind und welche Auswirkungen diese Changes haben. Das CAB bewertet dann nicht die Anforderung, sondern die sich daraus ergebenden Changes. Die ChGM liefert die Struktur, die dem PMI und vielen klassischen Projektmethoden fehlt. Sie operationalisiert den

Zusammenhang zwischen Zielzustand und Anpassungen am As-Is und macht die n:n-Beziehung zwischen Requirements und Changes transparent.

Ein Beispiel für ein ChGM steht unter <https://geswo.com/en/drtortuga/chgm-change-governance-matrix/> zum Download bereit. Es kategorisiert Changes in 11 Changesdomänen, 120 Changeskategorien und 1.200 Changestypen. Außerdem enthält es ein Beispiel für ein mögliches RACI-Schema sowie 85 prägnante Rollenbeschreibungen in fünf Sprachen (Deutsch, Englisch, Französisch, Portugiesisch, Spanisch). Die folgende Tabelle veranschaulicht die 11 Changesbereiche:

Tabelle2 : Darstellung der 11 Changesdomänen der Change Governance Matrix (ChGM)

Change Domain	CD	Description (EN)
Technical Infrastructure	TI	System setup, transport paths, connectivity, monitoring, and platform operations
Processual Changes	PC	Business process logic, customizing, workflows, and functional adjustments
Organizational Structure	OS	Roles, escalation paths, communication logic, governance layers
Security & Compliance	SC	Authorizations, certificates, audit logging, data protection
Data Migration	DM	Master data loads, delta logic, validation, fallback strategies
Interfaces & Integration	IN	Middleware, IDocs, APIs, external system connectivity
UI/UX & User Guidance	UX	Fiori apps, SAP GUI, personalization, launchpad configuration
Hypercare & Support	HS	Monitoring, support structure, escalation logic, post-go-live stabilization
Transport Management	TM	CTS+, ChaRM, release bundling, transport approvals
Training & Enablement	TE	Training plans, user guides, quick reference materials, onboarding
Software Development / WRICEF	DE	Workflows, Reports, Interfaces, Conversions, Enhancements, Forms

Bitte beachten Sie, dass die Change-Governance-Matrix und ihr Konzept ausführlich in dem Buch „SAP Cutover Management: Matrix, Methoden und Best Practices, auch für Nicht-SAP-Projekte“ von Guido Nageldinger (2026) beschrieben sind. Das Buch ist bei Amazon und Lulu erhältlich.

2.8. Fazit

Requirements definieren den Sollzustand (To-Be). Changes definieren die Abweichung vom Istzustand. Die Projektplanung konzentriert sich auf Changes, da diese die eigentliche Arbeit darstellen. Bei der UAT werden Requirements getestet, da dabei der Sollzustand akzeptiert wird. Das PMI betrachtet Changes als ScopesChanges und unterscheidet nicht zwischen AnforderungsChanges und LösungsChanges. Die Change Governance Matrix (ChGM) geht noch einen Schritt weiter: Sie regelt Changes auf Lösungsebene und nutzt eine Zuordnung von Requirements zu Changes, um die n:n-Beziehung zwischen dem Sollzustand und den Ist-Anpassungen transparent und überschaubar zu machen. Dies ermöglicht es dem CAB, seine vorgesehene Rolle zu erfüllen: die Bewertung und Steuerung von Changes auf Lösungsebene.

Kapitel 3

Changes vs. Deliverables – Warum eine Sardinenfabrik offenbart, was im Projektmanagement oft übersehen wird



Abbildung5 : Vergessen Sie nicht, sich den dritten Teaser anzusehen, der auf der Zeitschriftenseite dieser Ausgabe aufgeführt ist!
<https://geswo.com/de/journal/issue01-2026/>

In der Projektmanagement-Literatur wird viel Zeit darauf verwendet, Requirements, Changes, Arbeitspakete und Liefergegenstände/Deliverables zu diskutieren. Doch eine entscheidende Unterscheidung fehlt fast überall: die klare Trennung zwischen **Requirements**, **Changes**, **Deliverables** und dem **Product Scope**. Erst wenn diese Konzepte klar voneinander getrennt sind, wird sichtbar, wie Projekte funktionieren – und warum Governance-Modelle in der Praxis oft Schwierigkeiten bereiten.

Um diese Unterscheidung anschaulich zu machen, nehmen wir ein einfaches Beispiel: eine Sardinenfabrik, wie im Teaser-Video gezeigt.

Stellen Sie sich eine kleine Fabrik mit acht Mitarbeitern und einer Maschine vor, die fünf Dosen Sardinen pro Minute produziert. Das ist der As-Is. Der Eigentümer hat nun ein Ziel: Die Produktion soll auf zehn Dosen pro Minute gesteigert werden. Dieses Ziel beschreibt den To-Be – und das ist eine **Anforderung**. Eine Anforderung definiert, wie die Welt in Zukunft aussehen soll, ohne jedoch festzulegen, wie man dorthin gelangt.

Die Frage, wie der Sollzustand erreicht werden soll, führt zur **Changesentscheidung**. Eine Change ist weder eine Arbeitseinheit noch ein Deliverableobjekt. Es handelt sich um eine **Führungsentscheidung**, die festlegt, wie der As-Is verändert wird, um den To-Be zu erreichen. In der Sardinenfabrik besteht die Change aus drei Elementen: Die Fabrikhalle wird erweitert, eine zweite Maschine wird angeschafft und acht zusätzliche Mitarbeiter werden eingestellt. Diese drei Maßnahmen bilden zusammen die Change – die Entscheidung darüber, wie man von fünf auf zehn Dosen pro Minute kommt.

Doch wie integrieren wir unsere Deliverables? Offensichtlich stehen Deliverables in Zusammenhang mit Changes. Es gibt diese traditionelle Kette (1), in der Requirements zu Changes führen und Changes zu Deliverables

(1) Requirements -> Changes -> Deliverables

Alle diese Beziehungen in (1) sind m:n-Beziehungen. Das Problem von (1) besteht darin, dass es die im vorigen Kapitel vorgestellte Outcome Delivery Structure (ODS) außer Acht lässt. Eine Besonderheit der ODS ist, dass dieselbe Menge an Changes mit unterschiedlichen ODS realisiert werden kann. Betrachten wir die Anforderung „Verdopplung der Sardinenproduktion“. Die Menge aller Changes C besteht aus den folgenden Elementen: c_1 = „Logistikkapazitäten verbessern“, c_2 = „Eine neue Maschine kaufen“ und $c_{(3)}$ = „Mehr Arbeitskräfte einstellen“.

(2) $C = \{c_1, c_2, c_3, \dots, c_n\}$

Für Ihre Changesmenge C können verschiedene Outcome Delivery Structures ods_j vorliegen. Oder anders ausgedrückt: Aus einer Changesmenge C wird eine bestimmte Outcome Delivery Structure ods_j gebildet.

(3) $ods_j = ods_j(C)$

Je nach ODS können völlig unterschiedliche Deliverables d_k vorliegen, auch wenn sie sich auf dieselbe Change beziehen. Nehmen wir an, wir untersuchen die Umsetzung der Change c_1 = „Logistikkapazitäten verbessern“. Für eine solche Change kann es mehr als eine ODS geben, zum Beispiel:

ods_1 : Bau eines Lagers

- Baugenehmigung (d_1)
- Architekturpläne (d_2)
- Lager (d_3)
- Schlüsselübergabe (d_4)

ods_2 : Lagerhaltung auslagern

- Ausschreibung (d_1)

- Dienstleistungsvertrag (d_2)
- Integrationspartner (d_3)
- Geschäftsübertragung

ods₃ : Automatisiertes Distributionszentrum

- Software (d_1)
- Fördertechnik (d_2)
- Robotik (d_3)
- Schulung (d_4)
- Inbetriebnahme (d_5)

Definition: Deliverable/Liefergegenstand

Die Menge der für das Deliverable relevanten Changes ist nicht notwendigerweise die gesamte Menge C . Vielmehr handelt es sich um die Teilmenge Z der Changes, die innerhalb eines bestimmten ods_j implementiert werden. Es gilt Folgendes

$$(4) \ Z \subseteq C$$

Ein einzelnes Deliverable d_i ist eine Funktion der Menge der Changes Z , die innerhalb einer bestimmten Outcome Delivery Structure $Z(ods_j)$ umgesetzt werden, sowie der Outcome Delivery Structure selbst $ods_j(Z)$, wobei

$$(5) \ d_i = d_i(Z(ods_j), ods_j(Z))$$

- Deliverables entstehen weder direkt aus den Changes noch direkt aus der Outcome Delivery Structure. Sie entstehen aus dem Zusammenspiel beider.
- Changes definieren, was transformiert werden soll. Die ODS definiert, wie die Transformation strukturiert ist. Deliverables entstehen aus der Kombination beider.
- Gleichung (5) veranschaulicht zudem, dass jedes Deliverable genau zu einer Outcome Delivery Structure (ODS) gehört.

Wenn beispielsweise dasselbe Deliverable zwei verschiedenen ODSs zugeordnet ist, entsteht eine Governance-Anarchie. Sofort stellen sich Fragen wie: Wer finanziert es? Wer genehmigt es? Wer ändert es? Wer nimmt es ab? Dieser Konflikt tritt häufig zwischen Programmen und Projekten auf.

Unabhängig von der Einbeziehung der ODS – was viele Governance-Rahmenwerke nicht tun – führt die Umsetzung eines Change zu vielen **Deliverables**. Und genau hier verschwimmt die Unterscheidung oft. Ein Deliverable ist ein Artefakt, ein Dokument, ein physisches Objekt oder ein genehmigungsreifer

Output, der durch die Umsetzung eines Changes entsteht. Deliverables sind nicht die Entscheidung selbst; sie sind die konkreten Liefergegenstände, die sich aus dieser Entscheidung ergeben.

Die Erweiterung der Fabrikhalle führt zu Bauplänen, Genehmigungsanträgen, Ausschreibungsunterlagen, Verträgen, Sicherheitskonzepten und letztendlich zur neuen Halle selbst. Der Kauf der zweiten Maschine führt zu Bedarfsanalysen, Lieferantenvergleichen, Bestellungen, Verträgen, Installationsplänen, Prüfprotokollen und der Maschine als physischem Deliverable. Die Einstellung neuer Mitarbeiter führt zu Stellenbeschreibungen, Stellenanzeigen, Bewerberlisten, Arbeitsverträgen, Schulungsunterlagen und Einarbeitungsplänen.

Damit wird eines deutlich: **Eine Change führt zu vielen Deliverables**. Eine Change verändert den As-Is Zustand; Deliverables dokumentieren oder setzen diese Change um.

3.1 Wenn der Product Scope ins Spiel kommt

Und hier kommt der **Product Scope** ins Spiel. Der Product Scope umfasst weder Requirements noch Changes. Er umfasst lediglich die durch die Umsetzung von Changes und die zugehörigen ODS entstandenen Deliverables. Der Product Scope beschreibt, was am Ende tatsächlich vorliegt – die neue Halle, die zweite Maschine, die geschulten Mitarbeiter, die Betriebsgenehmigungen, die Prüfprotokolle und alle anderen Deliverables, die den neuen Produktionszustand ermöglichen.

An dieser Stelle wird eine weitere strukturelle Besonderheit sichtbar – eine, die bei fast jeder großen Transformationsinitiative auftritt, insbesondere bei SAP-Programmen. Die Entscheidung für eine Change wird oft sehr früh getroffen. Ein Vorstand beschließt, SAP einzuführen, Prozesse zu harmonisieren oder bestimmte Module zu implementieren. Diese Entscheidung ist eine Change, da sie den As-Is grundlegend verändert. Die konkreten Deliverables, die sich aus dieser Entscheidung ergeben werden, sind zu diesem Zeitpunkt jedoch noch völlig unbekannt. Sie werden erst später im Rahmen der Outcome Delivery Structure (ODS) ausgearbeitet – sei es in einem Projekt, einem Programm oder einem Portfolio.

Darüber hinaus stellt das ODS eine Governance-Entscheidung dar, die festlegt, wie eine Reihe von Changes in Deliverables umgesetzt wird. Die Entscheidung bezüglich der ODS ist eine Governance-Entscheidung zur Umsetzung. Die Entscheidung über den Change ist eine Governance-Entscheidung.

Die ODS erzeugt nicht direkt Deliverables. Deliverables entstehen aus der Umsetzung von Changes innerhalb des ausgewählten ODS. Aus diesem Grund enthält eine Projektcharta die Change-Entscheidungen, nicht jedoch die Deliverables. Deliverables entstehen erst, wenn Teams damit beginnen, die Change-Entscheidung in die Praxis umzusetzen. Requirements bleiben für viele Projektmitglieder oft unsichtbar, da sie selten klar dokumentiert und noch seltener klar kommuniziert werden. Die Change-Entscheidung ist hingegen sichtbar, da sie in der Charta aufgeführt ist. Die Deliverables werden erst später sichtbar, da sie durch die im Rahmen der ODS geleistete Arbeit entstehen.

Dadurch entsteht eine klare und lückenlose Kette: **Requirements definieren den To-Be. Changes definieren die Entscheidung darüber, wie dieser Zustand erreicht wird. Die ODS definiert, wie die Changes umgesetzt werden. Deliverables sind die konkreten Liefergegenstände sowohl der Entscheidungen im Zusammenhang mit dem ODS als auch der Changes. Und der Product Scope besteht ausschließlich aus diesen Deliverables.** Nur die Kombination dieser vier Ebenen ermöglicht den Übergang vom As-Is zum To-Be – und nur die Deliverables bilden den tatsächlichen Product Scope.

Die Sardinenfabrik veranschaulicht auf einfache Weise, was auch in komplexen Projekten gilt: **Changes sind Governance. Deliverables sind Output. Der Product Scope besteht aus Deliverables. Und nur wenn diese Ebenen klar voneinander getrennt sind, funktioniert das Projektmanagement wirklich.**

Viele Projekte scheitern, weil Changes ohne klare Requirements beschlossen werden und der Aufwand auf der Ebene der Change statt auf der Ebene der Deliverables geschätzt wird. **Changes sind Governance. Deliverables sind Output. Nur Deliverables generieren Aufwand, Risiko und Product Scope.**

3.2. Die Change Illusion

In vielen Projekten entsteht eine eigentümliche Illusion – eine, die erst sichtbar wird, wenn die Terminologie klar voneinander getrennt ist: die **Change Illusion**. Sie entsteht, wenn eine Change-Entscheidung als Scope verkauft wird, obwohl es sich lediglich um eine Steuerungsentscheidung handelt. Die Organisation beschließt beispielsweise: „Wir werden eine neue Fabrik bauen.“ Dieser Satz klingt nach Scope, fühlt sich wie Scope an und wird im Lenkungsausschuss als Scope präsentiert. In Wirklichkeit handelt es sich jedoch nur um eine Change – eine Entscheidung darüber, wie der As-Is verändert wird. Der Scope ergibt sich erst später innerhalb des ODS, wenn die Deliverables durch die Umsetzung der ausgewählten Changes ausgearbeitet werden.

Die Sardinenfabrik verdeutlicht dies besonders gut. Zunächst wird die Entscheidung getroffen, „eine neue Fabrik zu bauen“. Das ist die Change-Entscheidung. Doch sobald das Projekt beginnt, wird diese Entscheidung schrittweise konkretisiert: Aus der Fabrik wird eine neue Halle, dann eine zweite Maschine, dann mehr Personal, dann neue Prozesse, dann Sicherheitskonzepte und schließlich ein ganzes Bündel von Liefergegenständen. Diese schrittweise Konkretisierung wird oft mit der Ausarbeitung des Projekt-Scope verwechselt. In Wirklichkeit handelt es sich um die Konkretisierung der Changes: die schrittweise Erweiterung einer Change-Entscheidung, die später zu Liefergegenständen führt.

Das PMI spricht von „Progressive Elaboration“ (schrittweiser Ausarbeitung) und bezieht sich dabei in der Regel auf den Scope. In der Praxis wird jedoch etwas anderes ausgearbeitet/elaboriert: nicht der Scope, sondern der Change. Die Change-Entscheidung wird frühzeitig getroffen und in der Projektcharter dokumentiert. Sie wird als Scope verkauft, obwohl der Scope erst auf der Ebene der Deliverables entsteht. Die Deliverables selbst sind zu diesem Zeitpunkt noch völlig unbekannt. Sie

entstehen später innerhalb der Outcome Delivery Structure – sei es ein Projekt, ein Programm oder ein Portfolio.

Diese Besonderheit führt zu einem systemischen Problem: Der Aufwand wird auf der Ebene des Changes geschätzt, obwohl der Aufwand erst auf der Ebene der Deliverables existiert. Unternehmen schätzen den Aufwand für die „Vergrößerung der Halle“, nicht für die Baupläne, Genehmigungen, Ausschreibungen, Verträge, Sicherheitskonzepte und Zulassungen. Sie schätzen den Aufwand für „eine zweite Maschine“, nicht für die Bedarfsanalyse, den Lieferantenvergleich, die Bestellung, die Installation, die Tests und die Betriebsbereitschaft. Sie schätzen den Aufwand für „mehr Personal“, nicht für die Stellenbeschreibungen, Stellenanzeigen, Vorstellungsgespräche, Verträge, Schulungen und Einarbeitungspläne.

Dadurch entsteht die „Change Illusion“: die Überzeugung, dass die Change den Scope ausmacht. Der Scope ergibt sich jedoch erst später, wenn die Deliverables definiert sind. Und genau hier scheitern viele Projekte. Changes stimmen nicht mit den Requirements überein, da diese oft unsichtbar bleiben. Die Deliverables werden sowohl von den Requirements als auch von den Changes abgekoppelt, da sie erst viel später innerhalb der gewählten Outcome Delivery Structure ausgearbeitet werden. Bis dahin ist die ursprüngliche Absicht oft verwässert, missverstanden oder vergessen worden.

Kapitel 4

Die „Change Illusion“ in SAP-Projekten



Abbildung6 : Vergessen Sie nicht, sich den vierten Teaser anzusehen, der auf der Journalseite dieser Ausgabe aufgeführt ist!
<https://geswo.com/de/journal/issue01-2026/>

4.1. Was wir aus dem Clip lernen können

Der obige Clip veranschaulicht zahlreiche Aspekte, die typisch für eine „Change Illusion“ sind:

1. „Chef, wir haben die Entscheidung zum Change falsch verstanden!“

„Sie sagten, wir sollten *eine neue Fabrik bauen*. Aber niemand hat uns *genau* gesagt, was die Fabrik eigentlich produzieren soll. Also haben wir einfach alles gebaut, was uns eingefallen ist.“

→ **Kernaussage:** Change ≠ Scope. Ohne konkrete Vorgaben entsteht Chaos.

2. „Die Achterbahn war Teil der ‚Produktivitätssteigerung‘!“

„Wir dachten, wenn die Sardinen schneller rutschen, würden wir mehr produzieren. Also haben wir eine Achterbahn gebaut, damit die Dosen schneller durch die Halle sausen können.“

→ **Kernaussage:** Wenn Entscheidungen über Changes ohne Requirements getroffen werden, entstehen absurde Lösungen.

3. „Das Schwimmbecken war eine Sicherheitsmaßnahme!“

„Der Lenkungsausschuss sagte, wir bräuchten *mehr Sicherheit*. Also haben wir ein Becken gebaut, damit niemand hart auf dem Boden aufschlägt, wenn er aus der Achterbahn fällt.“

→ **Kernaussage:** Vage Vorgaben führen zu Deliverables, die nichts mit dem Ziel zu tun haben.

4. „Die Dosenboote waren ein Projekt zur Prozessharmonisierung!“

„Ihr habt gesagt, wir sollten die Prozesse harmonisieren. Wir wussten nicht, welche. Also haben wir aus den Dosen Boote gebaut, damit alle Mitarbeiter denselben Weg in ihre Pause nehmen.“

→ **Kernaussage:** Ohne klare Deliverables führt „Harmonisierung“ zu absurden Interpretationen.

5. „Die fliegenden Sardinen waren ein Innovations-Liefergegenstand!“

„Du hast gesagt, wir sollten innovativ sein. Also haben wir den Sardinen Flügel gebaut. Jetzt fliegen sie überall herum und rufen ‚Caipirinha!‘“

→ **Kernaussage:** Innovation ohne definierten Rahmen führt zu Chaos statt zu Fortschritt.

6. „Wir haben die Entscheidung zum Change schrittweise weiter elaboriert ... bis sie explodierten.“

„Du hast gesagt: *Neue Fabrik*. Daraus wurde dann: neue Halle → neue Maschine → neue Prozesse → neue Sicherheitskonzepte → neue Freizeitaktivitäten → neue Mitarbeiterzufriedenheit → neue Achterbahn → neues Schwimmbad → neue Schule für fliegende Sardinen.“

→ **Kernaussage:** Das ist die „**Change Illusion**“: Man arbeitet die Change aus, nicht den Scope.

7. „Wir hatten keine konkreten Deliverables, also haben wir uns einfach welche ausgedacht.“

„Niemand hat uns gesagt, was wir liefern sollten. Also haben wir einfach alles geliefert, was uns einfiel. Sogar die Sardinenboote. Und die Sardinen-Flugschule. Und die Wartungspläne für die Achterbahn.“

→ **Kernaussage:** Ohne definierte Deliverables gerät der Scope außer Kontrolle.

Definition der „Change Illusion“

Die „Change Illusion“ ist die falsche Annahme, dass eine Entscheidung zu einem Change automatisch den Scope definiert.

Sie tritt auf, wenn Organisationen eine übergeordnete Governance-Entscheidung („Wir brauchen eine neue Fabrik“, „Wir brauchen mehr Sicherheit“, „Wir brauchen Harmonisierung“) so behandeln, als hätte sie den tatsächlichen Scope bereits definiert.

In Wirklichkeit definiert eine Change nicht den Product Scope. Sie definiert lediglich die Entscheidung, den aktuellen Zustand zu verändern. Der Product Scope ergibt sich erst später durch die Deliverables, die innerhalb der gewählten Outcome Delivery Structure (ODS) ausgearbeitet werden.

Diese Illusion führt dazu, dass Organisationen Aufwand, Kosten und Risiken auf der Ebene der Changes statt auf der Ebene der Deliverables einschätzen. Teams diskutieren daher über Fabriken, Sicherheitsverbesserungen, Harmonisierungsinitiativen oder SAP-Implementierungen, lange bevor sie die Deliverables verstehen, die zu deren Realisierung erforderlich sind.

Infolgedessen werden Changes fälschlicherweise mit dem Scope verwechselt, Deliverables bleiben undefiniert, und die Projektplanung erfolgt auf der Grundlage von Annahmen statt anhand tatsächlicher Deliverableobjekte.

Die „Change-Illusion“ entsteht daher immer dann, wenn eine Change-Entscheidung mit den zu ihrer Umsetzung erforderlichen Deliverables verwechselt wird.

Changes sind Governance-Entscheidungen. Deliverables definieren den Product Scope. Die Verwechslung beider Begriffe führt zur „Change Illusion“.

4.2. Die „Change-Illusion“ in SAP-Projekten

Eine besonders reizvolle Form der „Change Illusion“ findet sich in SAP-Projekten. Es gibt sogenannte Requirement-Workshops, doch diese befassen sich selten mit tatsächlichen Requirements. Anstatt den Zielzustand zu definieren, springt die Diskussion direkt zur Umsetzung. Die Beteiligten klicken sich durch SAP-Transaktionen, vergleichen den „As-Is“ mit dem SAP-Standard, diskutieren Prozessvarianten und entscheiden, welche Konfigurationen, Erweiterungen oder Schnittstellen erforderlich sind. Das klingt nach Requirements, ist es aber nicht. Es handelt sich um die Ausarbeitung des Changes – die Ausarbeitung der Change-Entscheidung, die bereits in der Projektcharta skizziert ist.

Die SAP-Methodik geht somit über den eigentlichen Projekt-Scope hinaus. Requirements – also der angestrebte Geschäftszustand – werden selten dokumentiert und noch seltener kommuniziert. Deliverables – die Liefergegenstände, die den Zielzustand ermöglichen – werden erst später in der

Outcome Delivery Structure definiert. Was im Workshop definiert wird, sind Changes – keine Requirements und keine Deliverables. Kein Wunder, dass SAP-Projekte aus der Bahn geraten: Das Rahmenwerk wurde nie festgelegt. Das Unternehmen glaubt, Requirements zu definieren, definiert aber tatsächlich Changes. Und der Aufwand wird auf der Ebene der Changes geschätzt, obwohl Aufwand nur auf der Ebene der Deliverables existiert. Die „Change Illusion“ ist somit fest in der SAP-Methodik verankert – und erklärt, warum so viele Projekte scheitern, bevor sie überhaupt richtig begonnen haben.

SAP-Requirement-Workshops sind keine Requirement-Workshops. Es sind Change-Workshops, die als Requirement-Workshops getarnt sind.

Zugegeben, im Rahmen der aktuellen SAP-Activate-Methodik hat sich die Art und Weise, wie Requirements erfasst werden, grundlegend geändert. Früher gab es lange, traditionelle „Anforderungserfassungssitzungen“ (sessions for requirement gathering), um eine maßgeschneiderte Anforderungsspezifikation zu erstellen (Blueprint-Ansatz). Heute stützt sich SAP stark auf den Standard.

Der wichtigste und vorherrschende Workshop-Typ in der heutigen SAP-Welt ist der **„Fit-to-Standard“-Workshop**. Ergänzt wird dieser durch strategische Vorbereitungsworkshops und anschließende Detail-Sitzungen.

Ein Überblick über die wichtigsten Workshops:

4.2.1. „Fit-to-Standard“-Workshops

Ziel ist es hier, Prozesse an den SAP-Standard anzupassen. Die Hauptteilnehmer sind Bereichsleiter (Key-User) und Modulberater.

Diese Workshops finden während der Explore-Phase des Projekts statt. Der Standard wird vorgestellt, und das Unternehmen passt sich ihm an – nicht umgekehrt. Die Workshops werden nach Geschäftsbereich/SAP-Modul organisiert und verlaufen in der Regel wie folgt:

- **Durchlauf der Geschäftsprozesse:** Der Berater demonstriert vordefinierte Standardprozesse (SAP Best Practices) live auf einem funktionsfähigen Demosystem.
- **Eignungsprüfung:** Key-User aus den Fachabteilungen prüfen, ob ihre tägliche Arbeit durch diesen Standardprozess abgedeckt werden kann.
- **Delta-/Lückenanalyse:** Wenn etwas nicht passt – das heißt, wenn eine Lücke besteht –, wird genau dokumentiert, warum der Standard unzureichend ist und welcher geschäftliche Bedarf eine Anpassung erfordert.

4.2.2. Spezielle Workshops für technische Requirements (RICEFW)

Ziel ist es hier, technische Lösungen für Lücken (Gaps) zu entwerfen. Die Hauptteilnehmer sind Prozessverantwortliche und ABAP-/Technikberater.

Sobald in den „Fit-to-Standard“-Sitzungen Lücken oder Erweiterungen identifiziert werden, finden spezielle, vertiefende Workshops mit den IT- und technischen Beratern statt. Diese konzentrieren sich auf die sogenannten **RICEFW-Objekte**:

- **Berichts-Workshops:** Welche operativen Berichte oder analytischen Dashboards (z. B. unter Verwendung von SAP Fiori oder SAP Analytics Cloud) sind erforderlich?
- **Integrations-/Schnittstellen-Workshops:** Wie wird das SAP-System mit Drittsystemen (Measurement-Results, Webshops, HR-Tools) verbunden?
- **Formular-Workshops:** Abstimmung von Layouts und Pflichtfeldern für Rechnungen, Lieferscheine, Bestellungen oder Etiketten.
- **Erweiterungen / Eigenprogrammierung:** Besprechung von benutzerdefinierter Logik, die im SAP-Standardsystem nicht vorhanden ist.

4.2.3. Daten- und Migrations-Workshops

Ziel ist hier die Planung der Datenmigration aus dem Altsystemen. Die Hauptteilnehmer sind Datenanalysten und IT-Manager.

Ein SAP-System ist nur so gut wie seine Daten. Da die Datenmigration oft die größte Hürde darstellt, finden bereits frühzeitig spezielle Sitzungen statt:

- **Workshops zur Datenbereinigung und -zuordnung:** In diesen Sitzungen wird festgelegt, wie Altdaten (Kunden, Lieferanten, Materialstammsätze) bereinigt, transformiert und in die neuen SAP-Datenstrukturen (z. B. Geschäftspartner) übertragen werden.

4.2.4. Funktionsübergreifende und organisatorische Workshops

Ziel ist es hier, die Grundlage für das System zu schaffen. Die wichtigsten Teilnehmer sind das Projektmanagementteam und die Unternehmensarchitekten.

- **Workshops zur Organisationsstruktur:** Gleich zu Beginn muss festgelegt werden, wie das Unternehmen im System abgebildet wird (Buchungskreise, Werke, Verkaufsorganisationen, Lagerorte).
- **Workshops zu Berechtigungen und Sicherheit:** Wer darf später welche Transaktionen und Daten einsehen? Hier werden Rollen und Profile (z. B. für Einkäufer, Buchhalter, Lagerarbeiter) definiert.

4.3. Definition des Projekt-Scopes (Scopesausarbeitung, Scope Elaboration) anstelle der Definition der Deliverables (Elaboration of Deliverables)

Das Problem bei diesen Workshops besteht darin, dass sie eher Changes als Deliverables erarbeiten und somit die Definition des eigentlichen Product Scopes aufschieben.

4.3.1. Warum „Fit-to-Standard“-Workshops Changes definieren

Auf den ersten Blick scheinen „Fit-to-Standard“-Workshops klassische Requirement-Workshops zu sein, doch aus methodischer Sicht tun sie etwas ganz anderes. Sie definieren nicht den angestrebten Geschäftszustand, sondern untersuchen vielmehr, inwieweit der bestehende As-Is an den SAP-Standard angepasst werden kann. Der Fokus liegt nicht auf der Frage „Was wollen wir erreichen?“, sondern auf der Frage „Wie müssen wir unseren aktuellen Zustand ändern, damit er mit dem Standard übereinstimmt?“ Dies beschreibt nicht den Zielzustand, sondern die Change am aktuellen Zustand – also eine Change. Die Diskussion dreht sich um Prozessvarianten, Systemverhalten, Konfigurationsmöglichkeiten und Abweichungen vom Standard. Sobald eine Lücke identifiziert wird, ergibt sich automatisch ein Change: eine Entscheidung darüber, wie der As-Is angepasst werden soll. „Fit-to-Standard“ ist daher keine Anforderungsdefinition, sondern vielmehr die systematische Ausarbeitung von Changes, die sich aus der Entscheidung „Wir führen SAP ein“ ergeben.

4.3.2. Warum technische RICEFW-Workshops Changes ausarbeiten

Auch die technischen Workshops, die sich mit RICEFW-Objekten befassen, beschäftigen sich nicht mit Requirements, sondern mit der konkreten Ausgestaltung der zuvor im „Fit-to-Standard“-Prozess identifizierten Changes. **Ein Anforderungsmerkmal würde den geschäftlichen Nutzen beschreiben, den ein Bericht, eine Schnittstelle oder ein Formular bieten soll.** In den RICEFW-Workshops liegt der Fokus jedoch nicht auf dem Nutzen oder dem Zielzustand, sondern auf der technischen Umsetzung: Datenfelder, Tabellen, Funktionsbausteine, Mapping-Logik, Triggerpunkte und benutzerdefinierter Code. Dies ist reine Change-Ausarbeitung. Die Workshops klären, wie die zuvor getroffenen Change-Entscheidungen technisch umgesetzt werden sollen. Sie definieren nicht die geschäftlichen Requirements, sondern die technischen Changes am System. Somit sind RICEFW-Workshops ein Paradebeispiel dafür, wie SAP-Projekte direkt in die Umsetzung einsteigen, ohne jemals den angestrebten Geschäftszustand zu formulieren.

4.3.3. Warum und in welchen Workshops werden Changes ausgearbeitet

Migrations-Workshops erscheinen oft als geschäftsorientierte Sitzungen, doch auch sie dienen der Ausarbeitung von Changes. Die Migration selbst ist eine Change: die Entscheidung, den aktuellen Zustand der Daten zu verändern, um sie in das neue System zu übertragen. Die Workshops definieren nicht die geschäftlichen Requirements an die Daten, sondern vielmehr, wie die vorhandenen Daten transformiert werden müssen, damit sie in SAP funktionieren. Der Schwerpunkt liegt auf der Zuordnung, der Bereinigung, der strukturellen Angleichung und der technischen Übertragung. Dies ist keine Definition von Requirements, sondern vielmehr die Ausarbeitung einer Change, die bereits in der

Charta festgehalten ist: „Wir migrieren unsere Daten nach SAP.“ Die Workshops klären die Umsetzung dieser Change, nicht den angestrebten Geschäftszustand der Datenqualität oder der Datenprozesse. Somit sind Migrationsworkshops ebenfalls Teil der Ausarbeitung von Changes – und nicht Teil der Anforderungserfassung.

4.3.4. Warum organisatorische Workshops Changes konkretisieren

Auf den ersten Blick scheinen die Organisationsworkshops – die sich auf Unternehmensstrukturen, Berechtigungen und Sicherheitskonzepte konzentrieren – strategischer und technischer Natur zu sein. Doch auch sie definieren keine Requirements, sondern vielmehr die strukturellen Changes, die erforderlich sind, um SAP überhaupt betreiben zu können. Die Definition von Buchungskreisen, Werken, Vertriebsorganisationen oder Rollenprofilen ist keine Beschreibung eines Zielzustands, sondern vielmehr eine Entscheidung darüber, wie der aktuelle Zustand des Unternehmens im System abgebildet werden soll. Diese Entscheidungen verändern die Organisation – und sind somit Changes. Die Workshops klären nicht, was das Unternehmen aus geschäftlicher Sicht erreichen will, sondern wie es sich strukturell verändern muss, um SAP nutzen zu können. Auch hier wird kein Product Scope definiert; stattdessen liegt der Fokus auf der Formulierung der Changesentscheidung: „Wir führen SAP ein.“ Die aus diesen strukturellen Changes resultierenden Deliverables werden erst später sichtbar – in Form von Organisationsmodellen, Rollenmatrizen, Sicherheitskonzepten und Systemkonfigurationen.

Die meisten SAP-Projekte scheitern nicht daran, dass die Requirements schlecht definiert sind. Sie scheitern daran, dass Changes mit Requirements verwechselt und Deliverables mit dem ProjektScope verwechselt werden.

4.4. Ein Vorprojekt als Lösung?

4.4.1. Warum eine zweite Aufwandsschätzung unbedingt notwendig ist

Erst nach den Workshops ergibt sich erstmals ein realistisches Bild - der tatsächlichen Changes. Erst jetzt weiß das Projekt, welche Prozessanpassungen, Erweiterungen, Schnittstellen, Datenmigrationen und organisatorischen Changes wirklich notwendig sind. Die ursprüngliche Aufwandsschätzung basiert jedoch fast immer auf der Change-Entscheidung aus der Projektcharta – also auf einer groben Vorstellung wie „Wir führen SAP ein“ oder „Wir harmonisieren Prozesse“. Diese Schätzung ist zwangsläufig ungenau, da sie nicht auf konkreten Deliverables basiert. Eine zweite Aufwandsschätzung wäre daher nicht nur sinnvoll, sondern methodisch unerlässlich: Erst nach der Workshop-Phase und der Ableitung der Deliverables liegen tatsächlich die Informationen vor, die für eine korrekte Aufwandsermittlung erforderlich sind. Ohne diese zweite Schätzung plant das Projekt auf der Grundlage einer Illusion – und stößt an eine Wand, weil die tatsächlichen Deliverables nie bewertet wurden.

4.4.2. Warum ein Vorprojekt die einzig sinnvolle Lösung wäre

Ein Vorprojekt würde genau das leisten, was die SAP-Methodik auslöst: Es würde den Rahmen schaffen. Es würde den angestrebten Geschäftszustand definieren, die Deliverables identifizieren, die Changes bewerten und die Outcome Delivery Structure vorbereiten. Ein Vorprojekt wäre die Phase, in der die „Change Illusion“ zerstreut wird und die Deliverables sichtbar werden. Erst dann könnte ein Projekt ordnungsgemäß geplant werden. In der Praxis wird dieser Schritt jedoch fast immer übersprungen. Anstatt ein Vorprojekt durchzuführen, stürzen sich die Beteiligten direkt in die Workshops – und damit direkt in die Phase der Changesausarbeitung. Das Projekt beginnt mit der Lösung, noch bevor das Problem definiert wurde. Ein Vorprojekt wäre die methodisch korrekte Antwort auf die „Change Illusion“, wird jedoch selten durchgeführt, da es Zeit, Geld und Klarheit erfordert – drei Dinge, die in frühen Projektphasen oft als „verzögernde“ Faktoren wahrgenommen werden. Ein Vorprojekt würde den angestrebten Geschäftszustand definieren, die Requirements ermitteln, die Changes bewerten, die Deliverables ableiten und die Outcome Delivery Structure vorbereiten.

4.4.3. Warum diese Lösung systematisch abgelehnt wird

Die Ablehnung einer zweiten Aufwandsschätzung oder eines Vorprojekts ist kein Zufall, sondern ein strukturelles Muster. Organisationen wollen schnelle Entscheidungen, klare Budgets und feste Zeitpläne – lange bevor klar ist, was tatsächlich umgesetzt werden muss. Eine zweite Schätzung würde offenbaren, dass die ursprüngliche Planung unrealistisch war. Ein Vorprojekt würde zeigen, dass die Entscheidung zum Change nicht mit den Requirements übereinstimmt. Beides würde die Illusion zerstören, dass das Projekt „unter Kontrolle“ ist. Deshalb wird die methodisch korrekte Lösung abgelehnt: Sie würde Transparenz schaffen, wo bisher nur Hoffnung herrschte. Die Change Illusion ist angenehm, weil sie den falschen Eindruck von Planungssicherheit vermittelt. Die Realität der zu erbringenden Leistungen ist unangenehm, weil sie Aufwand, Risiko und Komplexität offenlegt. Genau deshalb wird die fundierte Lösung systematisch verhindert.

4.4.4. Warum Projekte ohne diese Lösung unweigerlich scheitern

Fehlt die zweite Aufwandsschätzung und wird kein Vorprojekt durchgeführt, basieren die Projektpläne auf der „Illusion der Change“. Changes werden als Scope verkauft, obwohl der Scope nur auf der Ebene der Deliverables existiert. Der Aufwand wird auf der Ebene der Changes geschätzt, obwohl sich der Aufwand nur aus den Deliverables ergibt. Risiken werden unterschätzt, weil die Deliverables unbekannt sind. Abhängigkeiten werden übersehen, da weder die Outcome Delivery Structure noch die daraus resultierenden Deliverables vollständig definiert wurden. Und die Umstellung verläuft chaotisch, da POD → OD nur funktioniert, wenn Deliverables vorhanden sind. Das Projekt scheitert nicht, weil die Beteiligten schlechte Arbeit leisten, sondern weil die Methodik den Rahmen nie festgelegt hat. Ohne die zweite Schätzung und ohne ein Vorprojekt bleibt das Projekt im Dunkeln – und gerät unweigerlich aus der Bahn, weil es nie wusste, wo der Rahmen tatsächlich lag.

4.4.5. Warum die Lösung nur funktioniert, wenn die Deliverables zunächst aus den ausgearbeiteten Changes abgeleitet werden

Damit eine zweite Schätzung und eine Zeitschätzung überhaupt funktionieren können, muss zwischen der Changesausarbeitung und der Planungsphase ein Schritt eingefügt werden – ein Schritt, der in SAP-Projekten fast immer fehlt: die Ableitung von Deliverables. Erst wenn klar ist, welche Deliverables sich aus den identifizierten Changes ergeben, kann ein Projekt realistisch eingeschätzt werden. Die Workshops liefern lediglich die Change-Entscheidungen – also die Beschreibung, wie der As-Is verändert werden soll. Aufwand, Risiko und Komplexität ergeben sich jedoch nicht aus der Change-Entscheidung selbst, sondern aus den Deliverables, die sich aus dieser Entscheidung ergeben. Eine Lücke im „Fit-to-Standard“ ist kein Scope-Element, sondern ein Hinweis darauf, dass ein Deliverable erstellt werden muss. Eine Erweiterung ist kein Aufwandsfaktor, sondern vielmehr ein Bündel von Deliverables wie Spezifikationen, Entwicklungsartefakte, Testfälle, Dokumentation und betriebliche Freigaben. Eine Datenmigration ist keine Aufgabe, sondern eine Vielzahl von Deliverables wie Zuordnungstabellen, Bereinigungsregeln, Transformationsskripte, Testläufe und Abnahmeberichte.

Solange diese Deliverables nicht sichtbar sind, befindet sich das Projekt im Dunkeln. Die Ausarbeitung von Changes erzeugt lediglich die Idee der Changes, nicht die Arbeit selbst. Erst die Ableitung der Deliverables offenbart, was tatsächlich erstellt, getestet, dokumentiert und übergeben werden muss. Ohne diesen Schritt ist jede Aufwandsschätzung zwangsläufig falsch, da sie auf einer Governance-Entscheidung basiert und nicht auf den Deliverables, die den Product Scope ausmachen. Die zweite Schätzung kann daher erst erfolgen, sobald die Deliverables klar definiert sind – und nicht vorher. Es handelt sich dabei nicht um eine „Korrektur“ der ersten Schätzung, sondern vielmehr um die erste realistische Schätzung überhaupt, da sie auf der Outcome Delivery Structure basiert und nicht auf der „Change Illusion“.

Genau deshalb wäre ein Vorprojekt so wertvoll: Es würde die Leistungen aus den Changes ableiten, bevor das eigentliche Projekt beginnt. Da dieser Schritt jedoch Zeit, Geld und Transparenz erfordert, wird er in der Praxis oft abgelehnt. Das Unternehmen verlangt frühzeitig Budgets, Zeitpläne und Zusagen, obwohl die Deliverables noch gar nicht bekannt sind. Dies hält die „Change Illusion“ aufrecht: Es wird ein Projekt geplant, dessen Scope nie definiert wurde. Die Lösung liegt daher auf der Hand: Zuerst die Deliverables aus den ausgearbeiteten Changes ableiten, dann Aufwand und Zeit abschätzen – und erst dann entscheiden, ob das Projekt in dieser Form überhaupt realisierbar ist.

4.5. Warum SAP Activate den wichtigsten Schritt der Projektplanung nicht berücksichtigt

4.5.1. SAP Activate ist eine Implementierungsmethodik, keine Methodik zur Festlegung von Deliverables.

SAP Activate wurde nie dafür konzipiert, den geschäftlichen Scope zu definieren. SAP Activate ist eine **Beschleunigungsmethodik**, die davon ausgeht, dass der Scope bereits klar ist – auch wenn dies in der Realität nie der Fall ist. Die Methodik setzt voraus, dass das Unternehmen weiß, was es erreichen will,

und dass sich die Deliverables bereits implizit aus dem SAP-Standard ableiten lassen. Das trifft jedoch nicht zu. SAP Activate geht von der Lösung (SAP-Standard) aus, nicht vom Problem (Anforderung). Deshalb fehlt der Schritt der Ableitung der Deliverables: Die Methodik geht davon aus, dass der Scope „im Standard verankert“ ist und nicht erst sichtbar gemacht werden muss. Dies ist ein methodischer Kurzschluss, der in der Praxis zur Scope Illusionen führt.

4.5.2. SAP Activate basiert auf der Annahme, dass der Standard den Scope definiert

Der Kern von SAP Activate ist „Fit-to-Standard“. Dies setzt implizit voraus, dass der SAP-Standard den Scope vorgibt. Wenn der Standard den Scope vorgibt, besteht theoretisch kein Bedarf an einer Ableitung von Deliverables – da der Scope ja bereits definiert ist. In der Realität ist der Standard jedoch nur ein Ausgangspunkt. Jede Abweichung führt zu Changes, und jede Change kann je nach der gewählten Outcome Delivery Structure zu einem oder mehreren Deliverables führen. SAP Activate überspringt diesen Schritt, da die Methodik davon ausgeht, dass die Abweichungen geringfügig sind und dass sich die Deliverables „von selbst ergeben“. Dies ist eine Illusion. In realen Projekten entstehen Hunderte von Deliverables, die niemals benannt, niemals geschätzt und niemals geplant werden. SAP Activate ignoriert diesen Schritt, da die Methodik den Scope im Standard verortet – nicht im Projekt.

4.5.3. SAP Activate vermeidet bewusst alles, was den Aufwand sichtbar macht

Ein weiterer Grund ist politischer Natur: Die Festlegung von Meilensteinen würde sofort offenbaren, wie groß der Scope-Bereich des Projekts tatsächlich ist. Es würde zeigen:

- wie viele Erweiterungen umgesetzt werden müssen
- wie viele Testfälle erstellt werden
- wie viele Datenobjekte migriert werden
- wie viele Rollen definiert werden müssen
- wie viele Dokumente erstellt werden müssen

Mit anderen Worten: **Es würde den tatsächlichen Aufwand offenlegen.**

SAP Activate vermeidet diesen Schritt, da er die Illusion zerstören würde, dass SAP-Implementierungen schnell, schlank und „Best Practice“ sind. Die Methodik ist darauf ausgelegt, Projekte verkaufsfähig zu machen – nicht planbar. Die Ableitung von Leistungen würde die Komplexität offenlegen und damit die Vertriebslogik der SAP Activate-Methodik untergraben. Deshalb wird dieser Schritt systematisch ausgelassen.

4.5.4. SAP Activate konzentriert sich auf die Ausarbeitung von Changes statt auf die Definition von Deliverables

Der wichtigste Punkt ist methodischer Natur: SAP Activate konzentriert sich vollständig auf **die Ausarbeitung von Changes**.

„Fit-to-Standard“ ist die Ausarbeitung von Changes. RICEFW-Workshops sind die Ausarbeitung von Changes. Die Datenmigration ist die Ausarbeitung von Changes. Die Organisationsstruktur ist die Ausarbeitung von Changes.

Activate definiert, wie der As-Is verändert wird, bietet jedoch keinen formalen Mechanismus zur Ableitung der Deliverables, die letztendlich den Product Scope definieren.

Die Methodik verfügt über keine Struktur zur Lieferung von Deliverables. Sie enthält keine Liste von Deliverables. Sie hat keinen Product Scope. Sie befasst sich ausschließlich mit Changes.

Deshalb fehlt der Schritt der Ableitung von Deliverables: „SAP Activate“ ist eine Methodik, die sich ausschließlich mit Changes befasst – und Deliverables als „implizit vorhanden“ betrachtet, obwohl sie in der Realität erst sichtbar gemacht werden müssen.

Kurz gesagt:

SAP Activate überspringt die Ableitung von Deliverables, da die Methodik davon ausgeht, dass der Scope im Standardsystem bereits definiert ist, dass die Changes den Arbeitsaufwand bestimmen und dass sich die Deliverables daraus ganz natürlich ergeben. In der Realität ist dies jedoch falsch – und genau deshalb geraten SAP-Projekte aus der Bahn.

Glossar

Outcome Delivery Structure (ODS): Die Gesamtheit aller Strukturebenen, die der Erzeugung eines Deliverables (Liefergegenstandes) dient, wie beispielsweise ein Projekts, ein Programms, ein Portfolios oder ein Epics. Die ODS stellt eine Governance-Entscheidung dar, die festlegt, wie eine Reihe von Changes in Deliverables umgewandelt wird.

Deliverable (Liefergegenstand): Ein Deliverable ist ein greifbares Dokument oder Artefakt (z. B. der vollständige Umstellungsplan), das zur Abnahme bereit ist und versioniert sowie formell genehmigt wurde.

Deliverable (erweiterte Definition): Ein einzelnes Deliverable d_i ist eine Funktion der Menge der Changes Z , die innerhalb einer bestimmten Outcome Delivery Structure $Z(ods_j)$ implementiert wurden, sowie der Outcome Delivery Structure selbst $ods_j(Z)$, wobei

$$a. d_i = d_i(Z(ods_j), ods_j(Z))$$

Die für das Deliverable relevante Menge an Changes ist nicht notwendigerweise die gesamte Menge C . Vielmehr handelt es sich um die Teilmenge Z der Changes, die innerhalb einer bestimmten ods_j implementiert werden. Es gilt Folgendes

$$b. Z \subseteq C$$

- Deliverables ergeben sich weder direkt aus den Changes noch direkt aus der Outcome Delivery Structure. Sie ergeben sich aus dem Zusammenspiel beider.
- Changes definieren, was transformiert werden soll. Die ODS definiert, wie die Transformation strukturiert ist. Deliverables entstehen aus der Kombination beider.
- Gleichung a. veranschaulicht zudem, dass jedes Deliverable genau zu einer Outcome Delivery Structure (ODS) gehört.

Voroperatives Deliverable (POD, pre-operational deliverable) und operatives Deliverable (OD, operational deliverable): Ein (in die Produktion) übertragbares Deliverable ist ein voroperatives Deliverable (POD), solange es noch nicht in der Produktion ist. Während der Umstellung wird ein POD in ein OD, also ein operatives Deliverable, überführt.

Übergangs-Deliverable (TD, transitional deliverable): Übergangs-Deliverables sind temporäre Artefakte, die geschaffen werden, um den Übergang vom voroperativen Zustand in die operative Umgebung zu ermöglichen. Sie sind nicht Teil des Endprodukts, dienen jedoch als wesentliche Hilfsmittel während der Umstellungsphase (cutover).

Cutover: Der Cutover wandelt *voroperative Deliverables (Pre-Operational Deliverables, POD)* in *operative Deliverables (Operational Deliverables, OD)* um. Diese operative Umwandlung bezieht sich stets auf die ausgewählte *Outcome Delivery Structure (ODS)* und wird durch *Übergangsdeliverables (Transitional Deliverables, TD)* ermöglicht.

Die gewählte ODS erfordert nicht immer PODs und kann daher auf einen Cutover verzichten. Stattdessen wird die ODS mit einer formellen operativen Übergabe beendet – wie bei Bauprojekten.

PODs, ODs und TDs sind Teil des Product Scopes. TDs sind zu Beginn selten bekannt und werden während der Projektdurchführung ausgearbeitet (elaboriert), meist im Rahmen der *Go-Live-Strategie* vor der Realisierungsphase.

Die Definition zeigt, dass ein Cutover nicht lediglich projektspezifisch, sondern ODS-abhängig ist. Er kann daher je nach der Struktur, auf die er sich bezieht, *projektspezifisch*, *programmspezifisch* oder *portfoliospezifisch* sein. Nicht jede ODS erfordert einen Cutover – aber jeder Cutover erfordert eine ODS.

Die Definition setzt Cutover-Aktivitäten mit der Transformation von PODs in Verbindung. Ein Cutover ist daher nicht nur eine Ansammlung technischer Systemaktivitäten. Es handelt sich um ein soziotechnisches Ereignis, das sowohl systemische als auch geschäftliche Changes zum Ausdruck bringt.

Changes & Requirements: Requirements definieren den Zielzustand (To-Be). Changes definieren die Abweichung vom As-Is. Die Projektplanung konzentriert sich oft auf Changes, da diese fälschlicherweise mit der eigentlichen Arbeit verwechselt werden. Bei dem UAT werden Requirements getestet, damit der Zielzustand akzeptiert wird.

Change Illusion: Die Change Illusion ist die falsche Annahme, dass eine Change-Entscheidung automatisch den Scope definiert.

Sie tritt auf, wenn Organisationen eine übergeordnete Governance-Entscheidung („Wir brauchen eine neue Fabrik“, „Wir brauchen mehr Sicherheit“, „Wir brauchen Harmonisierung“) so behandeln, als hätte sie den tatsächlichen Scope bereits definiert.

In Wirklichkeit definiert eine Change nicht den Product Scope. Sie definiert lediglich die Entscheidung, den aktuellen Zustand zu verändern. Der Product Scope ergibt sich erst später durch die Deliverables, die innerhalb der gewählten Outcome Delivery Structure (ODS) ausgearbeitet werden.

Diese Illusion führt dazu, dass Organisationen Aufwand, Kosten und Risiken auf der Ebene der Changes statt auf der Ebene der Deliverables einschätzen. Teams diskutieren daher über Fabriken, Sicherheitsverbesserungen, Harmonisierungsinitiativen oder SAP-Implementierungen, lange bevor sie die Deliverables verstehen, die zu deren Realisierung erforderlich sind.

Infolgedessen werden Changes fälschlicherweise mit dem Scope verwechselt, Deliverables bleiben undefiniert, und die Projektplanung erfolgt auf der Grundlage von Annahmen statt anhand tatsächlicher Deliverables.

Die „Change-Illusion“ entsteht daher immer dann, wenn eine Changesentscheidung mit den zu ihrer Umsetzung erforderlichen Deliverables verwechselt wird.

Changes sind Governance-Entscheidungen. Deliverables definieren den Product Scope. Die Verwechslung beider Begriffe führt zur „Illusion der Change“.

Die Scope-Illusion bezeichnet die Annahme, dass der Projektumfang bereits feststeht, bevor die zu erbringenden Leistungen definiert wurden. Sie tritt auf, wenn Organisationen davon ausgehen, dass Anforderungen, Governance-Entscheidungen, Standards, Projektcharta oder Änderungen den Umfang bereits festlegen, obwohl die zu erbringenden Leistungen noch unbekannt sind.

Die „Change-Illusion“ stellt eine spezifische Ausprägung der allgemeineren „Scope-Illusion“ dar.

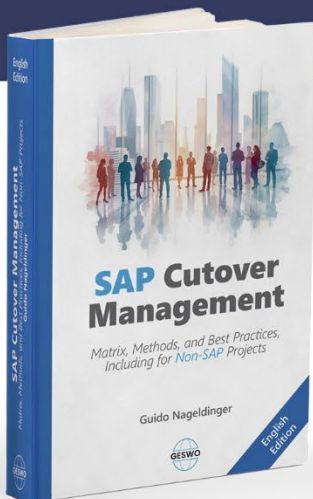
Das Konzept wird in einer zukünftigen Ausgabe des „Change-Illusion-Journals“ näher erläutert.

Guido Nageldinger PhD PMP
Independent S/4HANA Cutover & Release Advisor
Executive-Level Remote Engagement

BEFORE YOU APPROVE YOUR S/4HANA CUTOVER PLAN

Get an **independent** second opinion
before you lock budget, timeline, and
go-live risk.

GESWO Service Offering Cutover Coaching



Author of the book
SAP Cutover Management

**Book a 30 minute
executive briefing**

contact@geswo.com
<https://geswo.com/>

Typical Coaching Scenarios

- Strategy review
- Governance crisis
- Methodological input

Value Proposition

- Immediate solution to bottlenecks
- Methodological certainty instead of good feelings
- Minimal effort, maximum focus

Deliverable 1

Deep Dive Session (90 min.)

A facilitated, solution-oriented
one-on-one online session with a
GESWO senior expert.

Deliverable 2

Executive Summary (PDF)

A concise summary of the analyzed
root causes and the key points
discussed (approx. 1–2 pages).

Deliverable 3

Customized Action Plan

A concrete, prioritized set of
recommendations (Next Steps) for
direct implementation in your
project.

Deliverable 4

CoGM Method Shortcut

Relevant excerpts or checklists from
the Cutover Governance Matrix that
are precisely tailored to your specific
challenge (for internal use).



© 2026• GESWO – Editorial & Consulting, Guido Nageldinger • All Rights Reserved.
Registered as LLC in Wyoming, USA – Filing ID 2026-002064157.
RUC Paraguay: 8968931-3