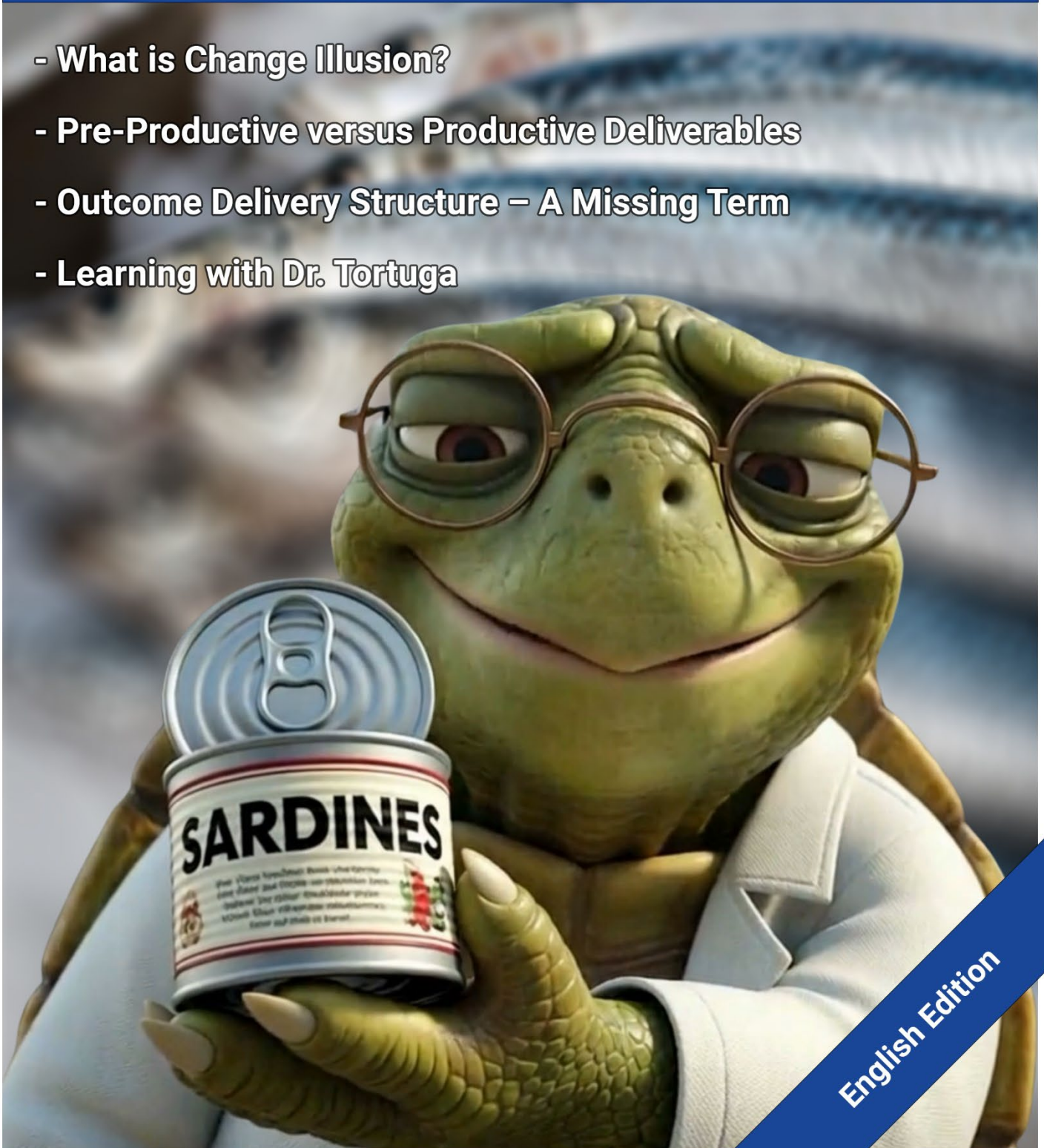


Change Illusion Journal

Project Management, Cutover Management & SAP Transformation

Advance Release

- What is Change Illusion?
- Pre-Productive versus Productive Deliverables
- Outcome Delivery Structure – A Missing Term
- Learning with Dr. Tortuga



English Edition



Editor's Note

Welcome to the first issue of the **Change Illusion Journal**. The “Change Illusion” represents a specific manifestation of the “Scope Illusion,” a broader concept. While the “Scope Illusion” describes the belief that scope exists before deliverables have been defined, the “Change Illusion” occurs when governance decisions are conflated with the Product Scope. Therefore, the term “**Scope Illusion**” is the central theme of this issue.

I have been involved in SAP projects for over 20 years, initially as an ABAP program, and subsequently as a project manager, program manager and cutover manager.

Nearly every project I have been involved with has made the same mistake. Projects rarely finished on time or within budget. Yes, there are many studies on project failure, and I am tired of quoting them. These studies appear every few months, but they don't nail it.

They don't identify the root cause of the problem, which you won't find in the PIM book or SAP Activate. I call this **Scope Illusion**. In this journal, I will define a few basics before guiding you to the root cause of poorly functioning SAP projects. Unfortunately, it is a methodological problem, which is why we need to examine the terminology.

Content

Editor's Note	2
Content	2
Chapter 1: From Outcome Delivery Structure (ODS) to Operational (OD) and Pre-Operational Deliverables (POD)	4
Chapter 2: Requirements, Changes, and the Change Governance Matrix (ChGM)	12
Chapter 3: Changes vs. Deliverables – Why a Sardine Factory Reveals What Project Management Often Overlooks	16
Chapter 4: Change Illusion in SAP Projects	21
Glossary	31

Impressum

The Change Illusion Journal – Issue 1 (Fall 2026) Semiannual Digital Publication

Publisher GESWO Editorial San Antonio, Encarnación, Paraguay | **Editor-in-Chief** Guido Nageldinger | **Contact** www.geswo.com publisher@geswo.com | **ISBN** 978-99989-(to be added) | **ISSN** (to be added once assigned)

Publication Frequency Semiannual (twice per year) | **Copyright** © 2026 GESWO Editorial. All rights reserved. This journal is published exclusively as a digital PDF edition. No part of this publication may be reproduced or distributed without written permission. | **Disclaimer** No warranty is given regarding the accuracy, completeness, or timeliness of the content. | **Trademarks** SAP, SAP Activate, Jira, ITIL, TOGAF, PMI and other terms mentioned are registered trademarks of their respective companies. | **Production** Cover and graphics created with Canva. Translations supported by DeepL.

Scope Illusion – Why SAP Projects Fail

By Guido Nageldinger

Editors Note - This issue of the *Change Illusion Journal* expands upon the ideas first introduced in the blog series. It refines the conceptual understanding of the Cutover by linking it to the Outcome Delivery Structure (ODS) and the Product Scope. Through new visual models — *Cutover vs. Formal Handover* and *Cutover Deliverable Flow* — the Journal establishes a systemic perspective on transformation, showing that Cutover is not merely a technical event, but a sociotechnical transition embedded in organizational change. The revised definition presented here marks a significant evolution of the concept, positioning Cutover as a structural phenomenon that connects deliverables, governance, and operational readiness across project, program, and portfolio levels.

Abstract – This cover story examines why many SAP and transformation projects fail despite following established project management methodologies. It introduces the concepts of the Outcome Delivery Structure (ODS), Deliverables, Cutover, and the Change Illusion as a framework for understanding how requirements, changes, and deliverables relate to each other. The cover story is divided into four chapters. The first chapter introduces a variety of new terms, such as the Outcome Delivery Structure (ODS). 'Cutover' is defined as the transformation from pre-operational deliverables (PODs) to operational deliverables (ODs) within the context of the ODS. Transitional deliverables (TDs) are introduced as enablers of this transformation process. The second chapter highlights the difference between requirements and changes. Requirements define the target state ('to be'). Changes, on the other hand, define the deviation from the current state. Project planning focuses on changes because they represent the actual work. It is argued that changes are often confused with requirements. The Change Governance Matrix (ChGM) and the mapping of requirements to changes are introduced to make the n:n relationship between the target state and the as-is adjustments transparent and manageable. This allows the CAB to fulfil its intended role of evaluating and governing changes at the solution level. The third chapter examines the relationship of deliverables in more detail. Deliverables do not emerge directly from changes, nor from the outcome delivery structure. Rather, they emerge from the interaction between the two. Changes define what shall be transformed. The ODS defines how the transformation is structured. Deliverables emerge from this combination. The deductive equation also illustrates that each deliverable belongs to one and only one ODS. Chapter four introduces the concept of 'Change Illusion', which is the false assumption that a decision to change automatically defines the scope of work. This occurs when organizations treat a high-level governance decision ('We need a new factory', 'We need more security', 'We need harmonization') as if it had already defined the actual scope. In reality, change does not define product scope. It merely signifies the decision to transform the current state. The product scope emerges later through the deliverables elaborated within the selected Outcome Delivery Structure (ODS). Chapter 4 also examines various so-called SAP 'requirements workshops' and concludes that they are, in fact, change workshops. The SAP Activate Methodology omits the derivation of deliverables because it assumes that the scope has already been defined in the standard system and that the changes define the work, with the deliverables following naturally. In reality, this is incorrect — and this is precisely why SAP projects go off the rails. The paper concludes that many projects fail because they elaborate changes rather than deliverables. A preliminary project is therefore recommended to derive deliverables from the elaborated changes before implementation begins, thereby establishing the missing framework for planning, governance, and cutover management.

Chapter 1

From Outcome Delivery Structure (ODS) to Operational (OD) and Pre-Operational Deliverables (POD)



Figure 1: Don't forget to watch the first teaser listed on the journal page of this issue! <https://geswo.com/en/journal/issue01-2026/>

Every company must adapt to market changes and transform its business. How? Through projects, programs, and complex transformation initiatives. For decades, we've been talking about portfolio, program, project, and, more recently, epic. But one thing has been missing until now: a common umbrella term that logically groups these levels and describes how organizations structure their work to deliver an outcome.

This is accurately where a new term comes in:

Definition - Outcome Delivery Structure (ODS): The totality of all structural levels that serve to produce an outcome.

ODS is a modern, framework-neutral approach that doesn't replace the familiar levels but organizes them more clearly.

ODS thus covers:

- classic project structures (PMI, PRINCE2)
- agile structures (Scrum, SAFe)
- hybrid models (SAP Activate)
- service-oriented models (ITIL)

1.1. Classic hierarchical structures

Classic hierarchical structures include:

- **Portfolio, Level 1** – strategic collection of all initiatives
- **Program, Level 2** – coordinated group of related projects
- **Epic, Level 3** – large work unit broken down into projects
- **Project, Level 4** – clearly defined deliverable with a start and end date
- **Task, Level 5** – smallest operational unit

1.2. The Agile Hierarchy (SAFe & Scrum)

In a scaled agile environment (**SAFe**), the terms fit directly into the existing structure:

- **Epic (Portfolio/Solution/Program Level):** sits at the very top. It describes a large, strategic initiative (e.g., “Develop a new customer portal”).
- **Feature (Program Level):** An epic is broken down into features (e.g., “Login system” or “Invoice overview”).
- **User Story (Team Level):** Features are broken down into user stories for implementation (e.g., “As a customer, I want to be able to log in with my email address”).
- **Sprint (Time Frame):** A sprint is **not a hierarchical work unit**, but rather a time-based container (usually 2–4 weeks). Within this container, the team works through a selection of user stories.

Visualization of the SAFe hierarchy with time units:

1. **Portfolio / Solution** (Strategy)
2. ↳ **Epic** (months to quarters)
3. ↳ **Feature** (Fits into a PI / Program Increment, approx. 8–12 weeks)
4. ↳ **User Story** (Fits into a **Sprint**, a few days)

1.3. Classification in SAP Activate

SAP Activate links agile concepts with traditional project phases:

- **Epic:** Corresponds to a major **deliverable** within a *workstream* (e.g., in the *Application Design & Configuration* workstream).

- **User Story:** In the SAP context, this is often recorded in the **backlog** during the *Explore phase*. It refines the requirements and, at the lowest level, corresponds to the **tasks** that are implemented in the sprint.
- **Sprint:** Takes place primarily in the **Realize phase**. Here, the user stories prioritized in the backlog are configured and tested in iterative cycles (sprints).

1.4. Classification in PMI & ITIL

- **PMI (Classic & Hybrid):** An **Epic** corresponds to a broad phase or a large work package in the project structure plan (PSP/WBS). The **User Story** is the specific, detailed activity. The **Sprint** replaces the classic, rigid schedule sequence with short, iterative segments.
- **ITIL (Value Streams):** An epic and its user stories are the specific requirements that flow through a **value stream**. The implementation of user stories within a **sprint** is part of the *Design & Transition* or *Obtain/Build* activities, ultimately delivering value to the customer.

1.5. The Conceptual Gap in All Frameworks: POD, OD and TD

We have perfect terms for structural levels (*portfolio, program, epic, feature, project, user story, task*), but not a single good term for the state of the deliverable as long as it is not yet in production within the ODS.

I have already clearly defined deliverables and outcome types in my book on SAP Cutover Management.

Definition: Deliverable

A deliverable is a tangible document or artifact (e.g., the complete cutover plan) that is ready for acceptance, versioned, and formally approved.

Terminology Note:

Along with the term *deliverable*, the phrases **result type** and **outcome** are also frequently used. These concepts are not further elaborated upon within this journal contribution. A **result type** typically represents a less formal component that may contribute to one or more deliverables, such as meeting minutes, workshop results, action items, or decision records. An **outcome** may be understood as the realized state resulting from the implementation of one or more changes through a set of deliverables.

However, there are two types of deliverables in a project. Some relate to the **Project Scope** and others to the **Product Scope**, to use PMI terminology. The **Project Scope** encompasses all deliverables necessary to manage, steer, and complete the project itself. These include, for example, project plans, governance documents, communication structures, status reports, and training materials. The **Product Scope** encompasses the deliverables that make up the actual product or system.

Only the deliverables that relate to the Product Scope are generally transferable to production—that is, relevant to the cutover. One could be pedantic here and argue that some deliverables from the project scope might indeed be transferred, such as Lessons Learned.

Example: The System Integration Test (SIT) and User Acceptance Test (UAT) are deliverables within the Project Scope. Even though they are acceptance criteria for the cutover, they are not transferred. System configurations, new process flows, trained staff, and documents ready for production use are transferable deliverables.

And here is the missing term: Pre-Operational Deliverable (POD)

Definition: Pre-Operational Deliverable (POD) versus Operational Deliverable (OD)

A transferable deliverable is a Pre-Operational Deliverable (POD) as long as it is not yet in production. During the cutover, a POD is transferred to an OD, or Operational Deliverable.

Interestingly, not all projects require pre-operational deliverables. In construction projects such as buildings or bridges, the product becomes operational immediately upon completion. These projects therefore do not involve a technical cutover in the IT sense. Instead, they rely on formal handover activities such as key delivery or regulatory approval.

Hybrid cases blur the boundary. In shipbuilding, for example, sea trials and safety checks act as pre-operational deliverables, while the ceremonial launch and delivery to the shipping company represent a formal handover. Similarly, hydropower plants require extensive pre-operational deliverables (test runs, grid synchronization, safety validation) and also involve a formal handover to the operator. This is illustrated in the diagram (Cutover–Handover Matrix) and table below.

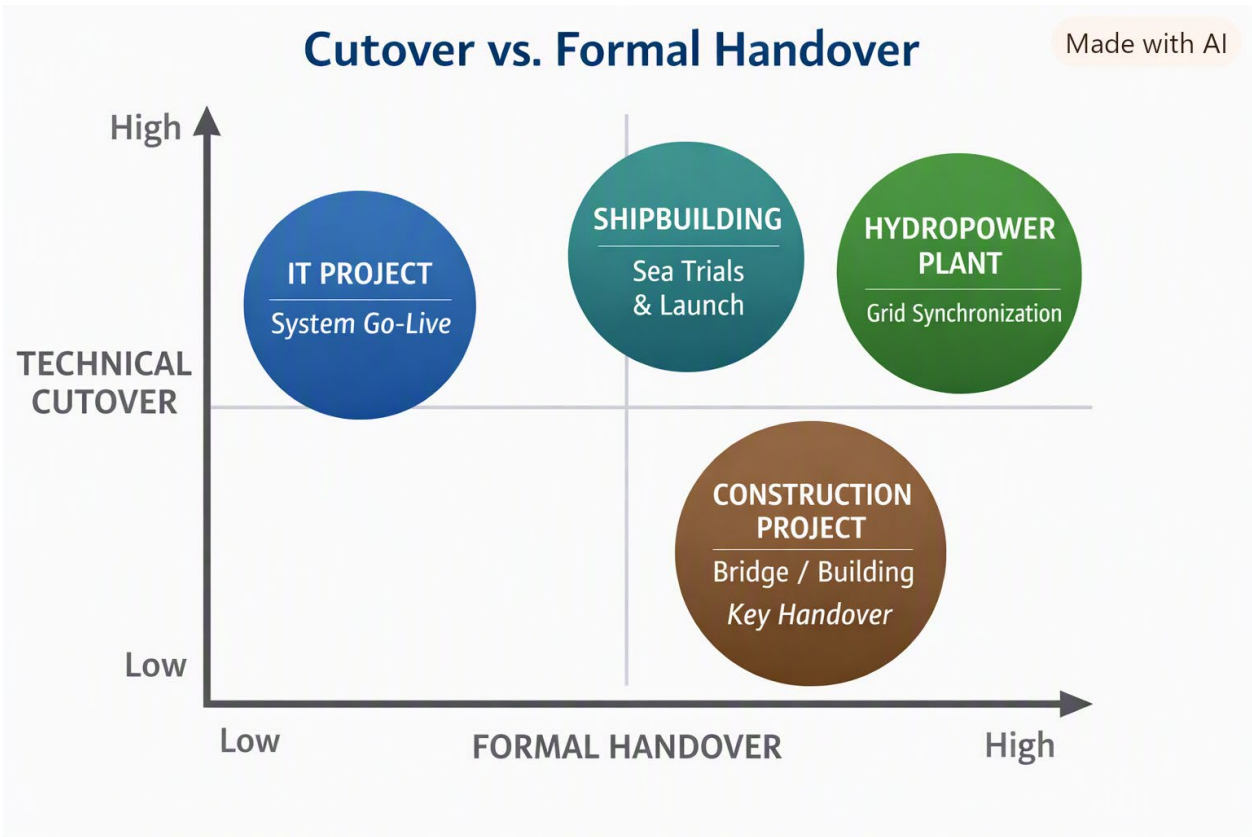


Figure 2: Cutover vs. Formal Handover The matrix illustrates how different project types balance technical cutover and formal handover. IT projects emphasize technical transitions, construction projects focus on formal delivery, while shipbuilding and hydropower plants combine both dimensions. Technical Cutover encompasses both system activation and organizational adaptation.

Tabelle 1: Comparison - Cutover Across Different Project Types

Project Type	Pre-Operational Deliverables (POD)	Nature of Cutover	Example of Handover	Remarks
IT Project	Yes, mandatory (e.g., test plans, migration scripts, data validation)	Technical-organizational, clearly defined cutover	Go-Live, system switch	Classic cutover scenario with POD → OD → Outcome
Construction Project (Bridge, Building)	No, no typical PODs	Formal handover, not technical	Key delivery, regulatory approval	Product is operational upon completion
Shipbuilding	Partial: dock tests, safety checks, crew training	Hybrid: technical and fromal cutover	Sea trials, delivery to shipping company	Boundary blurs: PODs exist, but handover also marks product completion
Hydropower Plant	Yes, extensive PODs (test runs, grid connection, safety checks)	Technical-organizational cutover with high complexity	Grid synchronization, commissioning	Closely resembles IT cutover: PODs are critical for safe operation

Technical Cutover refers to the coordinated transition of systems, processes, and organizational structures from a pre-operational to an operational state. While the term emphasizes the technical execution (system switch, data migration, configuration), it inherently includes organizational and procedural transformations required to sustain the new operational reality. Therefore, an IT cutover is not purely technical—it is a sociotechnical event that manifests both system and business change.

The Cutover–Handover Matrix introduces a new way to classify project transitions. It visualizes the continuum between technical activation and formal delivery across different industries, revealing that cutover is not purely technical but sociotechnical in nature.

Conclusion: Cutover is not a universal concept, but a spectrum. IT projects emphasize technical cutover, construction projects emphasize formal handover, and complex engineering projects (ships, power plants) combine both dimensions.

There is one more type of deliverable we haven't covered yet: Transitional Deliverables are, in fact, standalone deliverables, and they play a central role in the cutover context.

Definition: Transitional Deliverable (TD)

Transitional Deliverables are temporary artefacts created to enable the transition from the pre-operational state to the operational environment. They are not part of the final product but serve as essential enablers during the cutover phase.

Examples are:

- **IT Context:** Transition systems, temporary databases, migration scripts, sandbox environments.
- **Physical Projects:** Construction of a warehouse, temporary infrastructure, temporary solutions for testing or transition phases.
- **Organizational:** Training platforms, transition processes, temporary roles, or governance structures.

Transitional deliverables have the following characteristics:

- **Purpose:** Supports the transition, not the end product
- **Lifecycle:** Exist only for the duration of the cutover; decommissioned once operational stability is achieved.
- **Responsibility:** Often lies between the project and operations organizations
- **Value:** Reduce risk, maintain service continuity, and support readiness validation.

Cutover Deliverable Flow

Made with AI

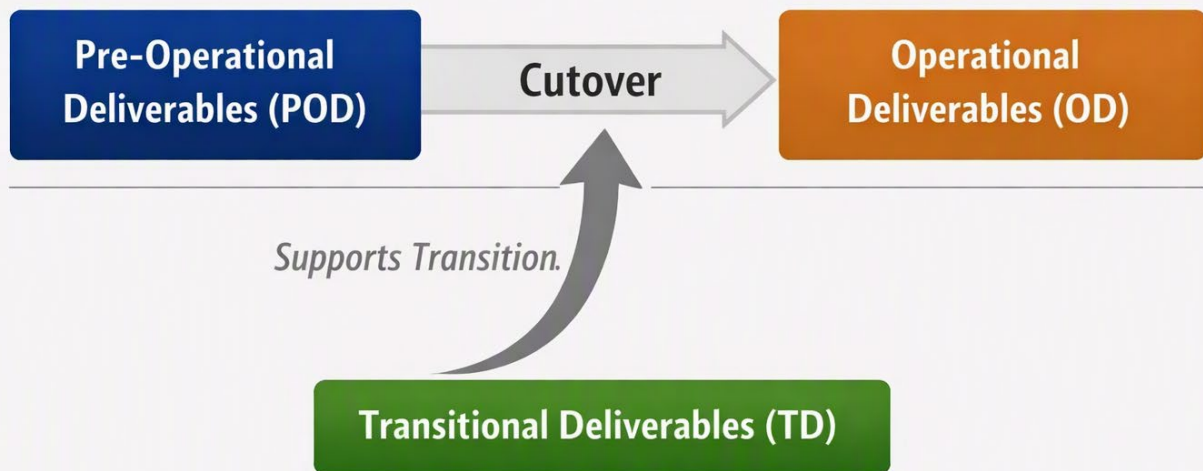


Figure 3: Transitional Deliverables (TD) are enabling artefacts that make the cutover possible. They do not transform into Operational Deliverables (OD) but provide the temporary structure, tools, or environments required for the transformation of Pre-Operational Deliverables (POD) into Operational Deliverables (OD). In this model, the TDs feed into the cutover process rather than being part of the transformation chain. Without them, the cutover could not occur safely or effectively.

While Pre-Operational Deliverables prepare the environment and Operational Deliverables define the final product, Transitional Deliverables ensure a controlled and stable passage between the two. They embody the temporary infrastructure of transformation — the scaffolding that allows the new operational reality to emerge safely, see Figure 2.

1.6. Transitional Deliverables and the Go-Live Strategy

Transitional Deliverables (TDs) belong to the product scope, even though they are rarely defined there. This distinction is essential to avoid the confusion between project management activities and product transformation activities. TDs are part of the product transition, not of project governance, and emerge during the elaboration of the **Go-Live Strategy**, which determines how Pre-Operational Deliverables (PODs) will be transformed into Operational Deliverables (ODs).

TDs are not part of the product itself but are essential enablers of the cutover. They represent temporary artefacts—such as migration environments, fallback systems, or provisional facilities—that make the transition technically and organizationally possible.

Because TDs are elaborated only when the Go-Live Strategy is defined, their existence argues for completing that strategy **before the realization phase** begins. Without a clear Go-Live Strategy, TDs

remain undefined, and the project risks entering realization without the necessary transitional infrastructure.

This distinction also helps to avoid the **Scope Illusion**: the misconception that project planning is complete when tasks are defined, rather than when deliverables—including transitional ones—are fully understood. TDs demonstrate that the product scope extends beyond the final product and includes the artefacts required to achieve operational readiness.

1.6. Summary

This chapter concludes with a definition of “cutover” and incorporates all the terms introduced so far.

Definition: Cutover

The Cutover transforms *Pre-Operational Deliverables (POD)* into *Operational Deliverables (OD)*. This operational transformation always refers to the selected *Outcome Delivery Structure (ODS)* and is **enabled by Transitional Deliverables (TD)**.

The chosen ODS does not always require PODs and therefore may not include a Cutover, concluding instead with a formal operational handover — as in building projects.

PODs, ODs, and TDs are part of the **product scope**. TDs are initially rarely known and are elaborated during project execution, most often within the *Go-Live Strategy* prior to the realization phase.

The definition also highlights that a Cutover is not merely project-specific but ODS-dependent. It may therefore be *project-specific*, *program-specific*, or *portfolio-specific*, depending on the structure it refers to. Not every ODS requires a Cutover — but every Cutover requires an ODS.

The definition relates Cutover activities to the transformation of PODs. A Cutover is therefore not just a collection of technical system activities. It is a **sociotechnical event** that manifests both system and business change.

TDs demonstrate that the Go-Live Strategy is not a late operational artifact but a strategic design activity that should be completed before realization begins. Their emergence during Go-Live planning reveals the hidden layer of product-scope deliverables that make the transformation from PODs to ODs possible — and thus expose the **Scope Illusion** (a term further elaborated in the subsequent chapter) when project planning is not deliverable-based.

Chapter 2

Requirements, Changes, and the Change Governance Matrix (ChGM)



Figure 4: Don't forget to watch the second teaser listed on the journal page of this issue! <https://geswo.com/en/journal/issue01-2026/>

2.1. Requirements and Changes – Two Perspectives on the Same Target State

In projects, “requirements” and “changes” are often mixed, even though they represent two different perspectives on the same target state. Requirements define the future state of a system or process. They describe what must be present in the target solution, regardless of the current as-is situation. Changes, on the other hand, define the deviation from the as-is state — the concrete adjustment needed to reach or modify the target state. This distinction is essential because it separates the functional definition of the target state from the technical or organizational work required to achieve it.

2.1.1. Requirements as the Definition of the Target State (To-Be)

A requirement describes what a system must deliver in the target solution. It typically emerges in early project phases and is solution-neutral. Requirements define scope, functional expectations and the criteria that will later be accepted in the UAT. They are not reactive but planning-oriented. A requirement such as “The system must support four languages” describes only the target state, not the work needed to reach it.

2.1.2. Changes as Adjustments to the As-Is State

A change always arises from the difference between the as-is state and the target state. If a system already supports Russian and Spanish (as-is) but the target solution requires four languages (to-be), the change consists of adding English and French. The change is therefore the concrete adjustment of the as-is state, not the abstract definition of the target state. Changes arise from new insights, technical constraints, regulatory requirements or defects and have direct impact on timeline, budget, testing and cutover.

2.2. Why UAT Tests Requirements — Not Changes

The User Acceptance Test (UAT) is the acceptance of the target state. It does not test the change itself, but the result of the change. Even if the system already supported Russian and Spanish, these languages must be tested again in the UAT because the UAT accepts the complete target state. The UAT therefore tests not only the new languages English and French, but all four languages, since the requirement defines four languages. The UAT is not a change test but a requirement test. It confirms that the target state has been fully achieved — regardless of how many changes were required.

2.2.1. Regression as an Integral Part of UAT

Retesting existing functionality is a logical consequence of target-state acceptance. Any change can affect existing functionality, whether through new configurations, new data models, new UI elements or new translation logic. The UAT does not test history; it tests the target state — and this target state includes all functionality, not only the changed parts.

2.3. Project Planning Focuses on Changes — Not Requirements

Requirements define the target state, but they do not generate work. Only the deviation from the as-is state — the change — generates effort, resource needs, timeline impact and budget. Therefore, projects do not plan “requirements”; they plan “changes.” Project planning answers the question: “What must we do to reach the target state?” And this question can only be answered based on changes, not requirements.

2.4. Why PMI Does Not Differentiate Here

In classical PMI logic, a change request is a modification of the approved scope. PMI views changes exclusively as scope changes and does not differentiate between requirement changes and solution changes. For PMI, every change is a modification of project scope, regardless of whether it is triggered by new functional requirements, technical constraints, regulatory demands or defects. This perspective is sufficient for traditional project management but does not reflect the complexity of modern solutions.

2.5. Requirement-to-Change Mapping — The Link Between Target (To-Be) and As-Is

In real projects, there is no linear relationship between requirements and changes. It is always an n:n relationship: one requirement can generate multiple changes, and one change can affect multiple requirements. A requirement may require no change if the as-is state already matches the target state. A change may arise without a requirement, for example due to regulatory demands or defects. Requirement-to-change mapping makes these relationships visible and provides the foundation for making changes traceable, testable and governable.

The mapping shows which concrete changes are necessary to implement a requirement and which requirements are affected by a change. It forms the basis for impact analysis, test planning and cutover evaluation. Without mapping, the connection between target state and as-is adjustments remains invisible — and so do risks, dependencies and regressions.

2.6. Why CAB Often Evaluates Requirements Instead of Changes

In ITIL, the Change Advisory Board (CAB) is intended to evaluate changes — the deviations from the as-is state needed to reach or adjust the target state. In practice, however, CAB meetings often present requirements instead of changes. The CAB then discusses functional target definitions, even though it should evaluate the impact of concrete changes. The reason is simple: requirement-to-change mapping is rarely performed. When it is unclear which changes arise from a requirement, the CAB has no choice but to evaluate the requirement itself.

As a result, the CAB loses its actual function as a solution-change governance body and becomes a scope-review board — similar to classical PMI logic. Risks, dependencies, regressions and cutover impacts remain invisible because the changes were not explicitly documented.

2.7. The Change Governance Matrix (ChGM™) as the Solution

The Change Governance Matrix (ChGM™) separates these layers clearly. It treats requirements as the definition of the target state and changes as deviations from the as-is state. Only through requirement-to-change mapping does it become visible which concrete changes are necessary to implement a requirement and what impacts these changes have. The CAB then evaluates not the requirement but the changes arising from it. The ChGM provides the structure that PMI and many classical project methods lack. It operationalizes the connection between target state and as-is adjustments and makes the n:n relationship between requirements and changes transparent.

An example of a ChGM is available for download on <https://geswo.com/en/drtortuga/chgm-change-governance-matrix/>. It categorizes changes into 11 change domains, 120 change categories and 1200 change types. It also provides an example of a possible RACI and 85 concise role descriptions in five languages (German, English, French, Portuguese, Spanish). The following table illustrates the 11 change domains:

Tabelle 2: Illustration of the 11 change domains of the 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

Please note that the Change Governance matrix and its concept are described in detail in the book *SAP Cutover Management: Matrix, Methods and Best Practices, Including for Non-SAP Projects* by Guido Nageldinger (2026). The book is available on Amazon and Lulu.

2.8. Conclusion

Requirements define the target state (to-be). Changes define the deviation from the as-is state. Project planning focuses on changes because they represent the actual work. UAT tests requirements because it accepts the target state. PMI views changes as scope changes and does not differentiate between requirement changes and solution changes. The Change Governance Matrix (ChGM) goes further: it governs changes at solution level and uses requirement-to-change mapping to make the n:n relationship between target state (to-be) and as-is adjustments transparent and manageable. This enables the CAB to fulfill its intended role: evaluating and governing changes at solution level.

Chapter 3

Changes vs. Deliverables – Why a Sardine Factory Reveals What Project Management Often Overlooks



Figure 5: Don't forget to watch the third teaser listed on the journal page of this issue! <https://geswo.com/en/journal/issue01-2026/>

Project management literature spends a great deal of time discussing requirements, changes, work packages, and results. Yet one crucial distinction is missing almost everywhere: the clear separation between **requirements**, **changes**, **deliverables**, and the **product scope**. Only when these concepts are cleanly separated does it become visible how projects function — and why governance models often struggle in practice.

To make this distinction tangible, let's use a simple example: a sardine factory, as shown in the tease video.

Imagine a small factory with eight employees and one machine producing five cans of sardines per minute. This is the as-is state. The owner now has a goal: production should increase to ten cans per minute. This goal describes the to-be state — and that is a **requirement**. A requirement defines how the world should look in the future, without specifying how to get there.

The question of how the to-be state will be achieved leads to the **change decision**. A change is not a work unit and not a result object. It is a **governance decision** that determines how the as-is state will be altered to reach the to-be state. In the sardine factory, the change consists of three elements: the

factory hall will be expanded, a second machine will be purchased, and eight additional employees will be hired. These three measures together form the change — the decision on how to move from five to ten cans per minute.

But how do we integrate our deliverables? Obviously, deliverables relate to changes. There is this traditional chain (1), where requirements derive changes and changes derive deliverables

(1) Requirements -> Changes -> Deliverables

All these relationships in (1) are m:n. The Problem of (1) is that it neglects the Outcome Delivery Structure (ODS) which was introduced in the previous chapter. A peculiarity of the ODS is that the same set of changes can be realized with different ODSs. Let's have a look at the requirement "Double sardine production". The set of all changes C consists of the following elements with c_1 = "Improve logistics capability", c_2 = "Buy a new machine" and c_3 = "Hire more workers".

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

For your set of changes C you can have different Outcome Delivery Structures ods_j . Or differently expressed – A particular Outcome Delivery Structures ods_j is created from a change set C.

(3) $ods_j = ods_j(C)$

Depending on the ODS – we can have completely different deliverables d_k even though they relate to the same change. Let's say we investigate realizing the change c_1 = "Improve logistics capability". For such a change more than one ODS can exist, for example:

ods₁: Build a Warehouse

- Building permit (d_1)
- Architectural plans (d_2)
- Warehouse (d_3)
- Key handover (d_4)

ods₂: Outsource Warehousing

- Request for Proposals (d_1)
- Service Provider Agreement (d_2)
- Integration Partner (d_3)
- Business Transfer

ods₃: Automated Distribution Center

- Software (d₁)
- Conveyor technology (d₂)
- Robotics (d₃)
- Training (d₄)
- Go-live (d₅)

Definition: Deliverable

The set of changes relevant to the deliverable is not necessarily the entire set C. Rather, it is the subset Z of changes that are implemented within a particular ods_j. The following applies

$$(4) Z \subseteq C$$

A single deliverable d_i is a function of the set of Changes Z implemented within a specific Outcome Delivery Structure Z(ods_j) and the Outcome Delivery Structure itself ods_j(Z), with

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

- Deliverables do not emerge directly from Changes nor directly from the Outcome Delivery Structure. They emerge from the interaction between both.
- Changes define what shall be transformed. The ODS defines how the transformation is structured. Deliverables emerge from the combination of both.
- Equation (5) also illustrates that every deliverable belongs to exactly one Outcome Delivery Structure (ODS).

If, for example, the same deliverable is owned by two different ODSs we have governance anarchy. You immediately get questions like: Who funds it? Who approves it? Who changes it? Who accepts it? This conflict arise frequently between programs and projects.

Independently of the inclusion of the ODS, which much governance framework don't do, implementing a change produces many **deliverables**. And this is where the distinction often becomes blurry. A deliverable is a **result object** — an artifact, a document, a physical object, or an approval-ready output created through the implementation of a change. Deliverables are not the decision itself; they are the concrete results that emerge from that decision.

Expanding the factory hall produces building plans, permit applications, tender documents, contracts, safety concepts, and ultimately the new hall itself. Purchasing the second machine produces needs analyses, supplier comparisons, purchase orders, contracts, installation plans, test protocols, and the

machine as a physical result. Hiring new employees produces job descriptions, advertisements, candidate lists, employment contracts, training materials, and onboarding plans.

This makes one thing clear: **one change produces many deliverables**. A change alters the state; deliverables document or implement that alteration.

3.1 When the Product Scope Enters the Picture

And this is where the **product scope** enters the picture. The product scope does not include requirements and does not include changes. It includes only the deliverables created through the implementation of changes and the relating ODS. The product scope describes what actually exists at the end — the new hall, the second machine, the trained employees, the operational approvals, the test protocols, and all other result objects that enable the new production state.

At this point, another structural peculiarity becomes visible — one that appears in almost every large transformation initiative, especially in SAP programs. The change decision is often made very early. A board decides to introduce SAP, harmonize processes, or implement certain modules. This decision is a change, because it fundamentally alters the as-is state. But the concrete deliverables that will emerge from this decision are still completely unknown at that moment. They are elaborated only later, inside the Outcome Delivery Structure (ODS) — whether a project, a program, or a portfolio.

Furthermore, the ODS represents a governance decision that determines how a set of Changes will be transformed into Outcomes. The decision related to the ODS is an implementation governance decision. The decision of the change is a business governance decision.

The ODS does not directly create deliverables. Deliverables emerge from the implementation of changes within the selected ODS. This is why a project charter contains the change decisions but not the deliverables. Deliverables emerge only when teams begin to operationalize the change decision. Requirements often remain invisible to many project members because they are rarely documented cleanly and even more rarely communicated clearly. The change decision, however, is visible because it appears in the charter. The deliverables become visible only later, because they arise through the work performed within the ODS.

This creates a clear and complete chain: **requirements define the to-be state. Changes define the decision on how that state will be reached. The ODS defines how the changes will be transformed. Deliverables are the concrete result objects of both decisions related to the ODS and the changes. And the product scope consists exclusively of those result objects.** Only the combination of these four layers enables the transition from as-is to to-be — and only the deliverables form the actual product scope.

The sardine factory shows in a simple way what is equally true in complex projects: **changes are governance. Deliverables are output. The product scope consists of deliverables. And only when these layers are clearly separated does project management truly work.**

Many projects fail because changes are decided without clear requirements, and effort is estimated at the change level instead of the deliverable level. **Changes are governance. Deliverables are work. Only deliverables generate effort, risk, and product scope.**

3.2. The Change Illusion

In many projects, a peculiar illusion emerges — one that becomes visible only when the terminology is cleanly separated: the **Change Illusion**. It arises when a change decision is sold as scope, even though it is merely a governance decision. The organization decides, for example, “We will build a new factory.” This sentence sounds like scope, feels like scope, and is presented as scope in the steering committee. But in reality, it is only a change — a decision about how the AS-IS state will be altered. The scope emerges later within the ODS, when deliverables are elaborated through the implementation of the selected changes.

The sardine factory makes this especially clear. First, the decision is made to “build a new factory.” That is the change decision. But as soon as the project begins, this decision is progressively elaborated: the factory becomes a new hall, then a second machine, then more staff, then new processes, then safety concepts, and finally an entire bundle of result objects. This progressive detailing is often mistaken for scope elaboration. In reality, it is change elaboration: the progressive expansion of a change decision that later leads to deliverables.

PMI speaks of “progressive elaboration,” usually referring to scope. But in practice, something else is elaborated: not the scope, but the change. The change decision is made early and documented in the charter. It is sold as scope, even though scope only emerges at the deliverable level. The deliverables themselves are still completely unknown at that point. They emerge later, within the Outcome Delivery Structure — whether a project, a program, or a portfolio.

This peculiarity leads to a systemic problem: effort is estimated at the change level, even though effort exists only at the deliverable level. Organizations estimate “expanding the hall,” not the building plans, permits, tenders, contracts, safety concepts, and approvals. They estimate “a second machine,” not the needs analysis, supplier comparison, purchase order, installation, testing, and operational readiness. They estimate “more staff,” not the job descriptions, advertisements, interviews, contracts, training, and onboarding plans.

This creates the Change Illusion: the belief that the change is the scope. But the scope emerges only later, when the deliverables are defined. And this is precisely where many projects fail. Changes do not match requirements because requirements often remain invisible. Deliverables become disconnected from both requirements and changes because they are elaborated only much later within the selected Outcome Delivery Structure. By then, the original intent has often been diluted, misunderstood, or forgotten.

Chapter 4

Change Illusion in SAP Projects



Figure 6: Don't forget to watch the fourth teaser listed on the journal page of this issue! <https://geswo.com/en/journal/issue01-2026/>

4.1. What We Can Learn from the Clip

The clip above illustrates numerous aspects that are typical of a “change illusion”:

1. “Boss, we misunderstood the change decision!”

“You said we were supposed to *build a new factory*. But nobody told us *exactly what* the factory was supposed to produce. So we just built everything we could think of.”

→ **Key message:** Change ≠ Scope. Without deliverables, chaos ensues.

2. “The roller coaster was part of the ‘productivity boost’!”

“We thought that if the sardines slid faster, we’d produce more. So we built a roller coaster so the cans could zoom through the hall faster.”

→ **Key message:** When change decisions are made without requirements, absurd solutions result.

3. “The pool was a safety deliverable!”

“The steering committee said we needed *more safety*. So we built a pool so no one would hit the ground hard when they fell off the roller coaster.”

→ **Key message:** Vague changes result in deliverables that have nothing to do with the goal.

4. “The can boats were a process harmonization project!”

“You said we should harmonize the processes. We didn’t know which ones. So we turned the cans into boats so that all employees would take the same route to their break.”

→ **Key message:** Without clear deliverables, “harmonization” leads to absurd interpretations.

5. “The flying sardines were an innovation deliverable!”

“You said we should be innovative. So we built wings for the sardines. Now they’re flying all over the place, shouting ‘Caipirinha!’”

→ **Key message:** Innovation without a defined scope leads to chaos instead of progress.

6. “We progressively elaborated on the change decision... until it exploded.”

“You said: *New factory*. Then it turned into: new hall → new machine → new processes → new safety concepts → new recreational activities → new employee satisfaction → new roller coaster → new pool → new flying sardine school.”

→ **Key message:** This is the “**change illusion**”: You elaborate on the change, not the scope.

7. “We didn’t have any deliverables, so we just made some up.”

“No one told us what we were supposed to deliver. So we just delivered everything we could think of. Even the sardine boats. And the sardine flight school. And the roller coaster maintenance schedules.”

→ **Key message:** Without defined deliverables, the scope spirals out of control.

Definition Change Illusion

The Change Illusion is the false assumption that a change decision automatically defines the scope of work.

It occurs when organizations treat a high-level governance decision ("We need a new factory", "We need more security", "We need harmonization") as if it already defined the actual scope.

In reality, a change does not define the product scope. It merely defines the decision to transform the current state. The product scope emerges later through the deliverables that are elaborated within the selected Outcome Delivery Structure (ODS).

This illusion causes organizations to estimate effort, cost, and risk at the change level instead of the deliverable level. Teams therefore discuss factories, security improvements, harmonization initiatives, or SAP implementations long before they understand the deliverables required to realize them.

As a result, changes are mistaken for scope, deliverables remain undefined, and project planning is performed on assumptions rather than actual result objects.

The Change Illusion therefore arises whenever a change decision is confused with the deliverables required to implement it.

Changes are governance decisions. Deliverables define the product scope. Confusing both creates the Change Illusion.

4.2. The Change Illusion in SAP Projects

A particularly charming form of the Change Illusion can be found in SAP projects. There are so-called requirements workshops, but they rarely address actual requirements. Instead of defining the target state, the discussion jumps straight to implementation. People click through SAP transactions, compare the "as-is" state with the SAP standard, discuss process variants, and decide which configurations, enhancements, or interfaces are necessary. This sounds like requirements, but it isn't. It is change elaboration—the elaboration of the change decision that is already outlined in the charter.

The SAP methodology thus goes beyond the actual scope of the project. Requirements—that is, the target business state—are rarely documented and even more rarely communicated. Deliverables—the output objects that enable the target state—are not defined until later in the Outcome Delivery Structure. What is defined in the workshop are changes—not requirements and not deliverables. No wonder SAP projects go off the rails: the framework was never established. The organization believes it is defining requirements, but in fact, it is defining changes. And effort is estimated at the change level,

even though effort only exists at the deliverable level. The “change illusion” is thus firmly anchored in the SAP methodology—and explains why so many projects fail before they’ve even really begun.

SAP requirements workshops are not requirements workshops. They are change workshops disguised as requirements workshops.

Okay, under the current SAP Activate methodology, the way requirements are gathered has fundamentally changed. In the past, there were long, traditional “requirements gathering” sessions to write a massive, customized requirements specification (blueprint approach). Today, SAP relies heavily on the standard.

The most important and dominant workshop type in today’s SAP world is the **Fit-to-Standard workshop**. This is complemented by strategic preparatory workshops and follow-up detailed sessions.

An overview of the most important workshops:

4.2.1. Fit-to-Standard Workshops

The goal here is to align processes with the SAP standard. The main participants are business unit managers (key users) and module consultants.

These workshops take place during the Explore phase of the project. The standard is presented, and the company adapts to it—not the other way around. The workshops are organized by business area/SAP module and typically proceed as follows:

- **Business Process Walkthroughs:** The consultant demonstrates predefined standard processes (SAP Best Practices) live on a functional demo system.
- **Fit Validation:** Key users from the business departments verify whether their daily work can be covered by this standard process.
- **Delta/Gap Identification:** If something does not fit—that is, if a gap exists—it is precisely documented why the standard is insufficient and what business need requires an adjustment.

4.2.2. Special Workshops for Technical Requirements (RICEFW)

The goal here is to design technical solutions for gaps. The main participants are process owners and ABAP/technical consultants.

As soon as gaps or extensions are identified in the “Fit-to-Standard” sessions, dedicated in-depth workshops are conducted with the IT and technical consultants. These focus on the so-called **RICEFW objects**:

- **Reporting workshops:** Which operational reports or analytical dashboards (e.g., using SAP Fiori or SAP Analytics Cloud) are necessary?

- **Integration/Interface Workshops:** How will the SAP system be connected to third-party systems (measurement results, web shops, HR tools)?
- **Forms Workshops:** Coordination of layouts and required fields for invoices, delivery notes, purchase orders, or labels.
- **Enhancements / Custom Code:** Discussion of custom logic that does not exist in the standard SAP system.

4.2.3. Data and Migration Workshops

The goal here is to plan the data migration from legacy systems. The main participants are data analysts and IT managers.

An SAP system is only as good as its data. Since data migration is often the biggest stumbling block, dedicated sessions are held early on:

- **Data Cleansing & Mapping Workshops:** These sessions define how legacy data (customers, vendors, material master records) is cleansed, transformed, and transferred into the new SAP data structures (e.g., business partners).

4.2.4. Cross-Functional & Organizational Workshops

The goal here is to lay the foundation for the system. The key participants are the project management team and the enterprise architects.

- **Organizational Structure Workshops:** Right at the start, it must be defined how the company will be represented in the system (company codes, plants, sales organizations, storage locations).
- **Authorization & Security Workshops:** Who will be allowed to view which transactions and data later on? Roles and profiles (e.g., for buyers, accountants, warehouse workers) are defined here.

4.3. Defining the project scope (scope elaboration) rather than defining the deliverables (deliverable elaboration)

The problem with these workshops is that they elaborate changes rather than deliverables and therefore postpone the definition of the actual product scope.

4.3.1. Why “Fit-to-Standard” Workshops Define Changes

At first glance, Fit-to-Standard workshops appear to be classic requirements workshops, but from a methodological perspective, they do something entirely different. They do not define the target business state, but rather examine to what extent the existing current state can be aligned with the SAP standard. The focus is not on “What do we want to achieve?” but on “How do we need to change our

current state so that it aligns with the standard?” This does not describe the target state, but rather the change to the current state—that is, a change. The discussion revolves around process variants, system behavior, configuration options, and deviations from the standard. As soon as a gap is identified, a change automatically arises: a decision on how the current state should be adjusted. “Fit-to-Standard” is therefore not a requirement definition, but rather the systematic elaboration of changes resulting from the decision “We are implementing SAP.”

4.3.2. Why Technical RICEFW Workshops Elaborate on Changes

The technical workshops that deal with RICEFW objects also do not address requirements, but rather the concrete design of the changes previously identified in the Fit-to-Standard process. **A requirement would describe the business benefit that a report, an interface, or a form is intended to provide.** In the RICEFW workshops, however, the focus is not on benefits or the target state, but on technical implementation: data fields, tables, function modules, mapping logic, trigger points, and custom code. This is purely change elaboration. The workshops clarify how the change decisions made earlier are to be technically implemented. They do not define the business requirements, but rather the technical changes to the system. Thus, RICEFW workshops are a prime example of how SAP projects jump straight into implementation without ever formulating the target business state.

4.3.3. Why and Workshops Elaborate Changes

Migration workshops often appear to be business-focused sessions, but they, too, elaborate on changes. The migration itself is a change: the decision to alter the current state of the data in order to transfer it to the new system. The workshops do not define the business requirements for the data, but rather how the existing data must be transformed so that it works in SAP. The focus is on mapping, cleansing, structural alignment, and technical transfer. This is not a definition of requirements, but rather the elaboration of a change that is already stated in the charter: “We are migrating our data to SAP.” The workshops clarify the implementation of this change, not the target business state of data quality or data processes. Thus, migration workshops are also part of change elaboration—and not part of requirements gathering.

4.3.4. Why Organizational Workshops Elaborate on Changes

At first glance, the organizational workshops—which focus on corporate structures, authorizations, and security concepts—appear to be strategic and technical in nature. However, they, too, do not define requirements, but rather the structural changes necessary to operate SAP in the first place. Defining company codes, plants, sales organizations, or role profiles is not a description of a target state, but rather a decision on how the company’s current state should be mapped in the system. These decisions change the organization—and are therefore changes. The workshops do not clarify what the company aims to achieve from a business perspective, but rather how it must change structurally in order to use SAP. Here, too, no product scope is defined; instead, the focus is on formulating the change decision: “We are implementing SAP.” The deliverables resulting from these structural changes

only become apparent later—in the form of organizational models, role matrices, security concepts, and system configurations.

Most SAP projects do not fail because requirements are poorly defined. They fail because changes are mistaken for requirements and deliverables are mistaken for scope.

4.4. A Preliminary Project as a Solution?

4.4.1. Why a Second Effort Estimate Is Absolutely Necessary

After the workshops, a realistic picture of the actual changes becomes available for the first time. Only now does the project know which process adjustments, enhancements, interfaces, data migrations, and organizational changes are truly necessary. However, the original effort estimate is almost always based on the change decision from the charter—that is, on a rough idea such as “We are implementing SAP” or “We are harmonizing processes.” This estimate is inevitably inaccurate because it is not based on deliverables. A second effort estimate would therefore not only be sensible but methodologically essential: Only after the workshop phase and the derivation of deliverables does the information necessary to correctly determine effort actually exist. Without this second estimate, the project plans based on an illusion—and hits a wall because the actual deliverables were never evaluated.

4.4.2. Why a Preliminary Project Would Be the Only Sound Solution

A preliminary project would do exactly what the SAP methodology skips: it would establish the framework. It would define the target business state, identify the deliverables, evaluate the changes, and prepare the Outcome Delivery Structure. A preliminary project would be the phase in which the “change illusion” is dispelled and the deliverables become visible. Only then could a project be properly planned. In practice, however, this step is almost always skipped. Instead of conducting a preliminary project, people jump straight into the workshops—and thus directly into the change elaboration phase. The project begins with the solution before the problem has been defined. A preliminary project would be the methodologically correct response to the “change illusion,” yet it is rarely carried out because it requires time, money, and clarity—three things that are often perceived as “delaying” factors in early project phases. A preliminary project would define the target business state, identify the requirements, evaluate the changes, derive the deliverables, and prepare the Outcome Delivery Structure.

4.4.3. Why This Solution Is Systematically Rejected

The rejection of a second effort estimate or a preliminary project is no coincidence, but rather a structural pattern. Organizations want quick decisions, clear budgets, and fixed schedules—long before it is clear what actually needs to be built. A second estimate would reveal that the original planning was unrealistic. A preliminary project would show that the change decision does not align with the requirement. Both would shatter the illusion that the project is “under control.” That is why the methodologically correct solution is rejected: it would create transparency where, until now, only hope

has existed. The illusion of change is comfortable because it gives the false impression of planning certainty. The reality of deliverables is uncomfortable because it reveals effort, risk, and complexity. That is precisely why the sound solution is systematically prevented.

4.4.4. Why Projects Inevitably Fail Without This Solution

If the second effort estimate is missing and no preliminary project is conducted, the project plans based on the “change illusion.” Changes are sold as scope, even though scope exists only at the deliverable level. Effort is estimated at the change level, even though effort arises only from deliverables. Risks are underestimated because the deliverables are unknown. Dependencies are overlooked because neither the Outcome Delivery Structure nor the resulting deliverables have been fully defined. And the cutover becomes chaotic because POD → OD only works if deliverables exist. The project fails not because people are doing a poor job, but because the methodology never established the framework. Without the second estimate and without a preliminary project, the project remains in the dark—and inevitably goes off the rails because it never knew where the framework actually lay.

4.4.5. Why the solution only works if deliverables are first derived from the elaborated changes

For a second and time estimate to work at all, a step must be inserted between the change elaboration and the planning phase—a step that is almost always missing in SAP projects: the derivation of deliverables. Only when it is clear which deliverables will result from the identified changes can a project be realistically assessed. The workshops merely provide the change decisions—that is, the description of how the current state is to be changed. However, effort, risk, and complexity do not arise from the change decision itself, but rather from the deliverables that result from that decision. A gap in the “Fit-to-Standard” is not a scope element, but an indication that a deliverable must be created. An enhancement is not a driver of effort, but rather a bundle of deliverables such as specifications, development artifacts, test cases, documentation, and operational approvals. A data migration is not a task, but rather a multitude of deliverables such as mapping tables, cleansing rules, transformation scripts, test runs, and acceptance reports.

As long as these deliverables are not visible, the project remains in the dark. Change elaboration generates only the idea of change, not the work itself. Only the derivation of deliverables reveals what actually needs to be built, tested, documented, and transferred. Without this step, any effort estimate is inevitably incorrect because it is based on a governance decision rather than on the deliverables that constitute the product scope. The second estimate can therefore only take place once the deliverables are clearly defined—and not before. It is not a “correction” of the first estimate, but rather the first realistic estimate of all, because it is based on the Outcome Delivery Structure and not on the “change illusion.”

This is precisely why a preliminary project would be so valuable: it would derive the deliverables from the changes before the actual project begins. But because this step requires time, money, and transparency, it is often rejected in practice. The organization wants budgets, schedules, and

commitments early on, even though the deliverables aren't even known yet. This perpetuates the “change illusion”: a project is planned whose scope has never been defined. The solution is therefore clear: first derive the deliverables from the elaborated changes, then estimate effort and time—and only then decide whether the project is even feasible in this form.

4.5. Why SAP Activate Does Not Include the Most Important Step in Project Planning

4.5.1. SAP Activate is an implementation methodology, not a deliverable methodology.

SAP Activate was never designed to define business scope. Activate is an **and acceleration methodology** that assumes the scope is already clear—even though in reality it never is. The methodology presupposes that the organization knows what it wants to achieve and that the deliverables can already be implicitly derived from the SAP standard. But that is not true. Activate starts with the solution (SAP standard), not with the problem (requirement). That is why the step of deriving deliverables is missing: The methodology assumes that the scope is “embedded in the standard” and does not first need to be made visible. This is a methodological short-circuit that leads to scope illusions in practice.

4.5.2. SAP Activate is based on the assumption that the standard defines the scope

The core of Activate is “Fit-to-Standard.” This implicitly assumes that the SAP standard dictates the scope. If the standard dictates the scope, then theoretically there is no need for deliverable derivation—since the scope would already be defined. But in reality, the standard is only a starting point. Every deviation generates changes, and every change may result in one or more deliverables depending on the selected Outcome Delivery Structure. Activate skips this step because the methodology assumes that the deviations are minor and that the deliverables “arise on their own.” This is an illusion. In real projects, hundreds of deliverables arise that are never named, never estimated, and never planned. Activate ignores this step because the methodology locates the scope in the standard—not in the project.

4.5.3. SAP Activate deliberately avoids anything that makes effort visible

Another reason is political: Deriving deliverables would immediately reveal just how large the project really is. It would show:

- how many enhancements need to be built
- how many test cases will be created
- how many data objects will be migrated
- how many roles need to be defined
- how many documents need to be created

In other words: **It would reveal the true effort involved.**

Activate avoids this step because it would shatter the illusion that SAP implementations are fast, lean, and “best practice.” The methodology is designed to make projects sellable—not plannable. Deriving deliverables would reveal the complexity and thus undermine the sales logic of the Activate methodology. That is why this step is systematically omitted.

4.5.4. SAP Activate Focuses on Change Elaboration Rather Than Deliverable Definition

The most important point is methodological: Activate is entirely focused on **change elaboration**.

“Fit-to-Standard” is change elaboration. RICEFW workshops are change elaboration. Data migration is change elaboration. Organizational structure is change elaboration.

Activate defines how the current state is changed but provides no formal mechanism for deriving the deliverables that ultimately define the product scope.

The methodology has no outcome delivery structure. It has no list of deliverables. It has no product scope. It deals only with changes.

That is why the step of deriving deliverables is missing: Activate is a methodology that exclusively elaborates on changes—and considers deliverables to be “implicitly present,” even though in reality they must first be made visible.

In short:

SAP Activate skips the derivation of deliverables because the methodology assumes that the scope is already defined in the standard system, that the changes define the work, and that the deliverables follow naturally. In reality, this is incorrect—and that is precisely why SAP projects go off the rails.

Glossary

Outcome Delivery Structure (ODS): The totality of all structural levels that serve to produce an outcome, such as a project, program, portfolio or epic. The ODS represents a governance decision that determines how a set of Changes will be transformed into Outcomes.

Deliverable: A deliverable is a tangible document or artefact (e.g. the complete cutover plan) which is ready for acceptance and has been versioned and formally approved.

Deliverable (advanced definition): A single deliverable d_i is a function of the set of Changes Z implemented within a specific Outcome Delivery Structure $Z(ods_j)$ and the Outcome Delivery Structure itself $ods_j(Z)$, with

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

The set of changes relevant to the deliverable is not necessarily the entire set C . Rather, it is the subset Z of changes that are implemented within a particular ods_j . The following applies

$$b. Z \subseteq C$$

- Deliverables do not emerge directly from Changes nor directly from the Outcome Delivery Structure. They emerge from the interaction between both.
- Changes define what shall be transformed. The ODS defines how the transformation is structured. Deliverables emerge from the combination of both.
- Equation a. also illustrates that every deliverable belongs to exactly one Outcome Delivery Structure (ODS).

Pre-Operational Deliverable (POD) and Operational Deliverable (OD): A transferable deliverable is a Pre-Operational Deliverable (POD) as long as it is not yet in production. During the cutover, a POD is transferred to an OD, or Operational Deliverable.

Transitional Deliverable (TD): Transitional Deliverables are temporary artefacts created to enable the transition from the pre-operational state to the operational environment. They are not part of the final product but serve as essential enablers during the cutover phase.

Cutover: The Cutover transforms *Pre-Operational Deliverables (POD)* into *Operational Deliverables (OD)*. This operational transformation always refers to the selected *Outcome Delivery Structure (ODS)* and is enabled by *Transitional Deliverables (TD)*.

The chosen ODS does not always require PODs and therefore may not include a Cutover, concluding instead with a formal operational handover — as in building projects.

PODs, ODs, and TDs are part of the product scope. TDs are initially rarely known and are elaborated during project execution, most often within the *Go-Live Strategy* prior to the realization phase.

The definition also highlights that a Cutover is not merely project-specific but ODS-dependent. It may therefore be *project-specific*, *program-specific*, or *portfolio-specific*, depending on the structure it refers to. Not every ODS requires a Cutover — but every Cutover requires an ODS.

The definition relates Cutover activities to the transformation of PODs. A Cutover is therefore not just a collection of technical system activities. It is a sociotechnical event that manifests both system and business change.

Changes & Requirements: Requirements define the target state (to-be). Changes define the deviation from the as-is state. Project planning often focuses on changes because they are mistaken for the actual work. UAT tests requirements because it accepts the target state.

Change Illusion: The Change Illusion is the false assumption that a change decision automatically defines the scope of work.

It occurs when organizations treat a high-level governance decision ("We need a new factory", "We need more security", "We need harmonization") as if it already defined the actual scope.

In reality, a change does not define the product scope. It merely defines the decision to transform the current state. The product scope emerges later through the deliverables that are elaborated within the selected Outcome Delivery Structure (ODS).

This illusion causes organizations to estimate effort, cost, and risk at the change level instead of the deliverable level. Teams therefore discuss factories, security improvements, harmonization initiatives, or SAP implementations long before they understand the deliverables required to realize them.

As a result, changes are mistaken for scope, deliverables remain undefined, and project planning is performed on assumptions rather than actual result objects.

The Change Illusion therefore arises whenever a change decision is confused with the deliverables required to implement it.

Changes are governance decisions. Deliverables define the product scope. Confusing both creates the Change Illusion.

Scope Illusion describes the belief that project scope exists before deliverables have been defined. It occurs when organizations assume that requirements, governance decisions, standards, project charters, or changes already define the scope, even though the deliverables remain unknown.

The Change Illusion represents one specific manifestation of the broader Scope Illusion.

The concept will be further elaborated in a future issue of the Change Illusion Journal.

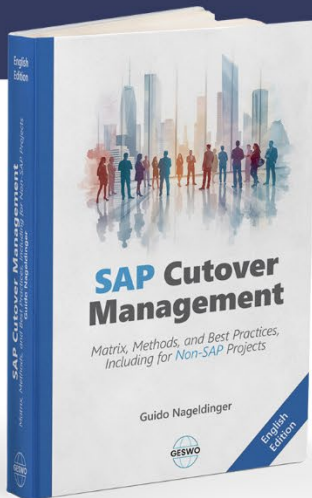
Guido Nageldinger PhD PMP

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